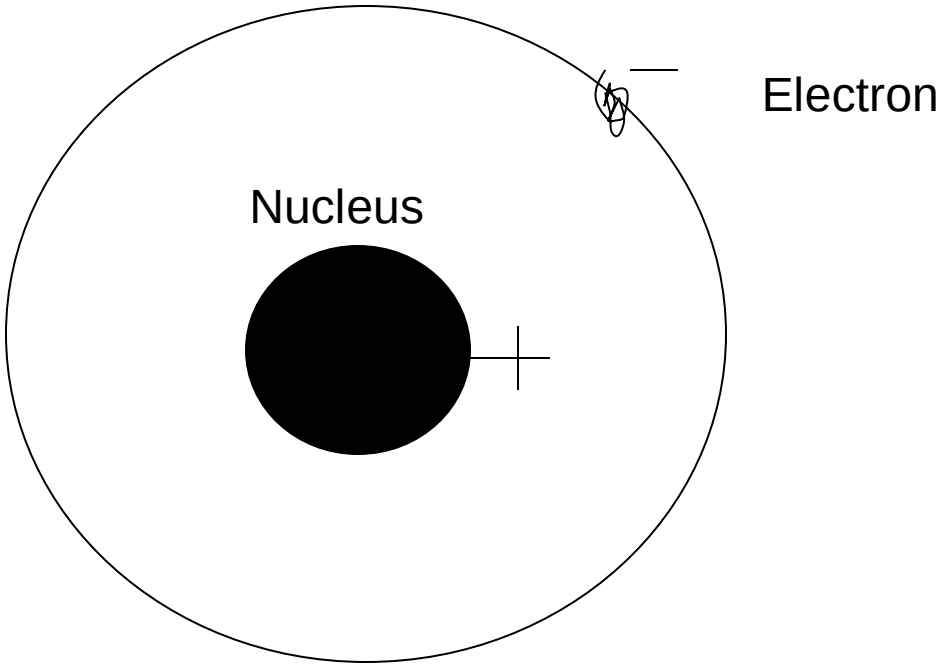
Chapter 5

Basic Electronics and Digital Logic

Digital Circuits

- *Combinational*: a circuit whose output depends only on its present inputs
- *Sequential*: a circuit whose output depends on its past as well as present inputs. That is, it depends on the *sequence* of inputs from the past up to the present.

Atom has a positively-charged nucleus and negatively-charged orbiting electrons



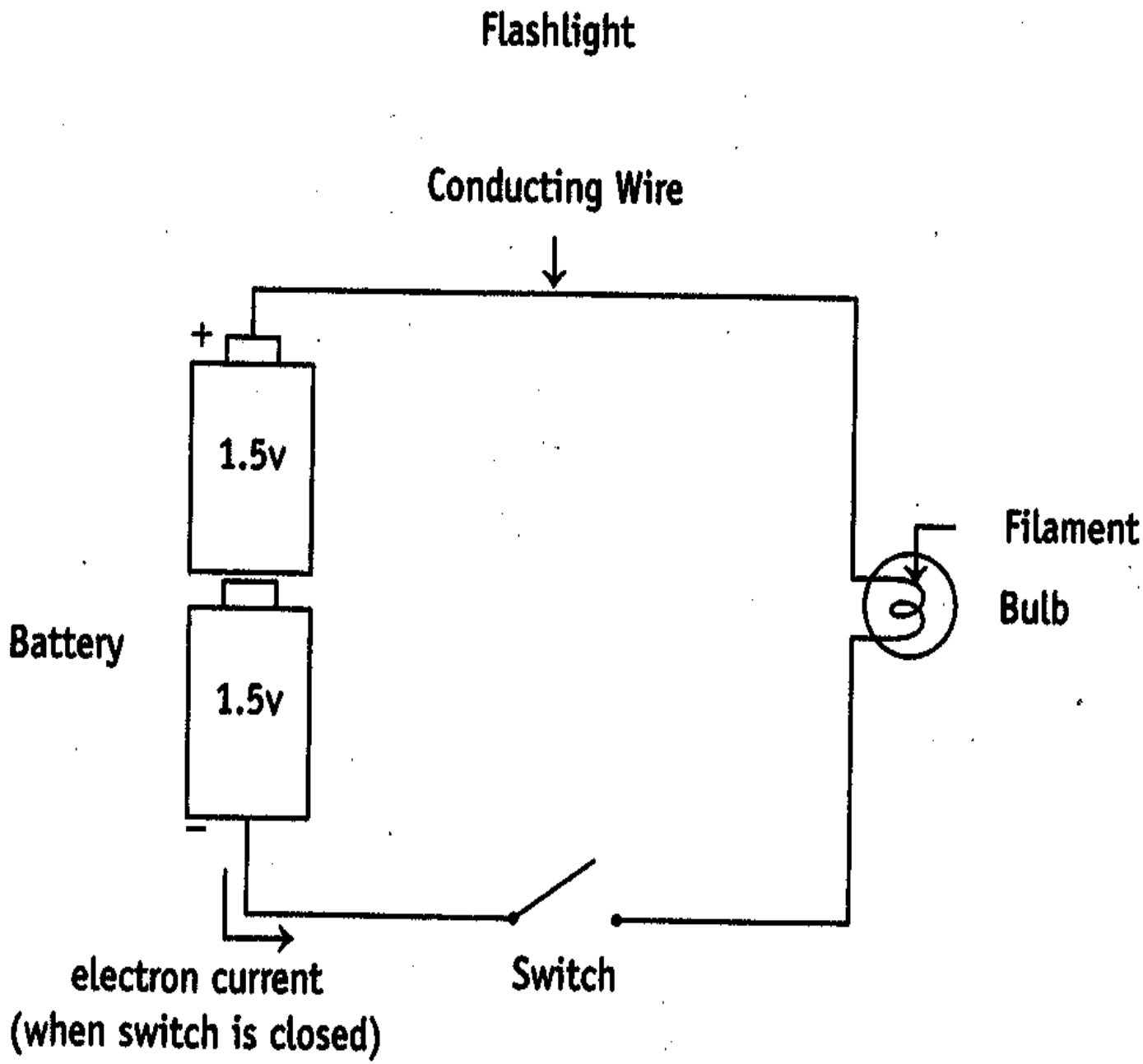
Like charges repel.
Unlike charges attract

A *conductor* has electrons that are not “stuck” to the nucleus.

An *insulator* has no “unstuck” electrons.

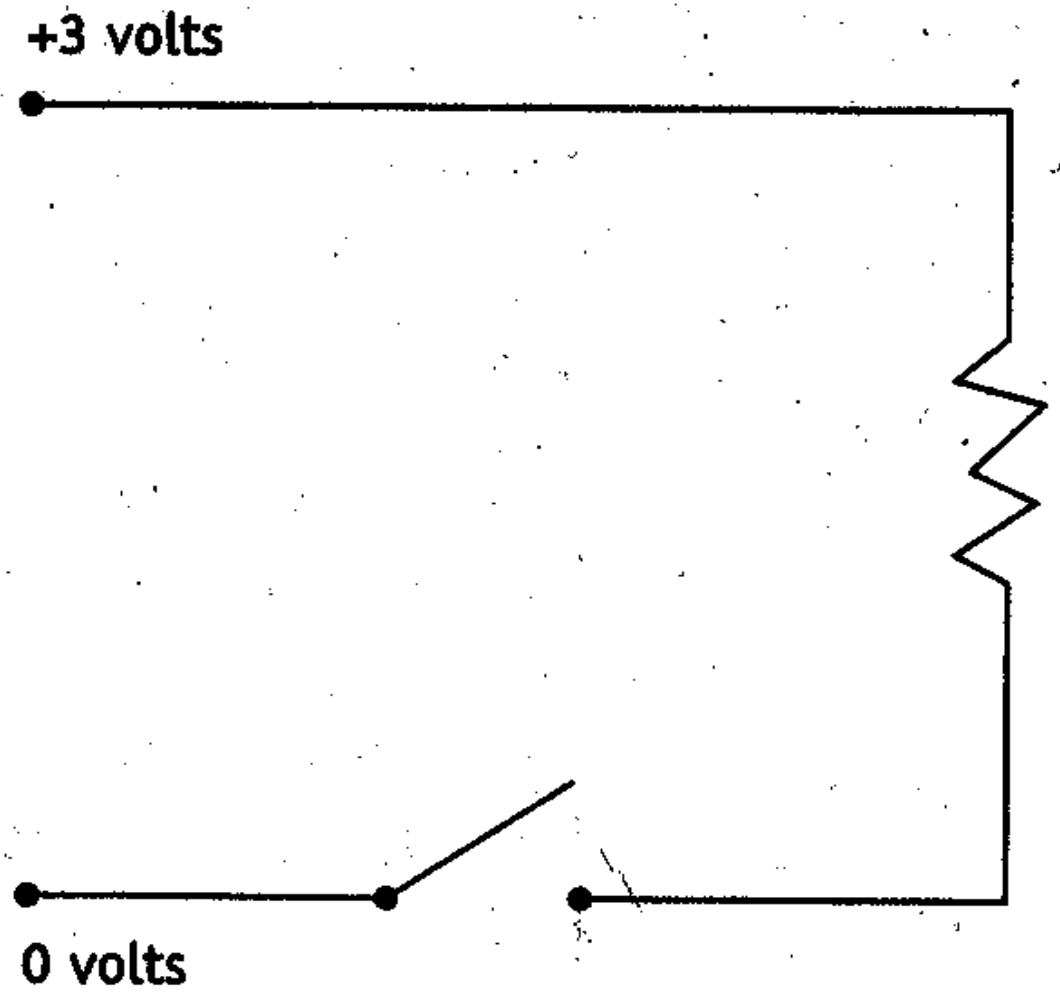
A flashlight is a simple electrical circuit

FIGURE 5.1


Flashlight
Conducting Wire
+
1.5v
Battery
1.5v
−
Filament
Bulb
electron current
(when switch is closed)
Switch

Electron current – flow of electrons
Voltage – provides “push” of electrons
Resistance – opposition to current

FIGURE 5.2 Schematic Diagram



Ohm’s law

I is current in amps
E is voltage in volts
R is resistance in ohms

$$I = E/R$$

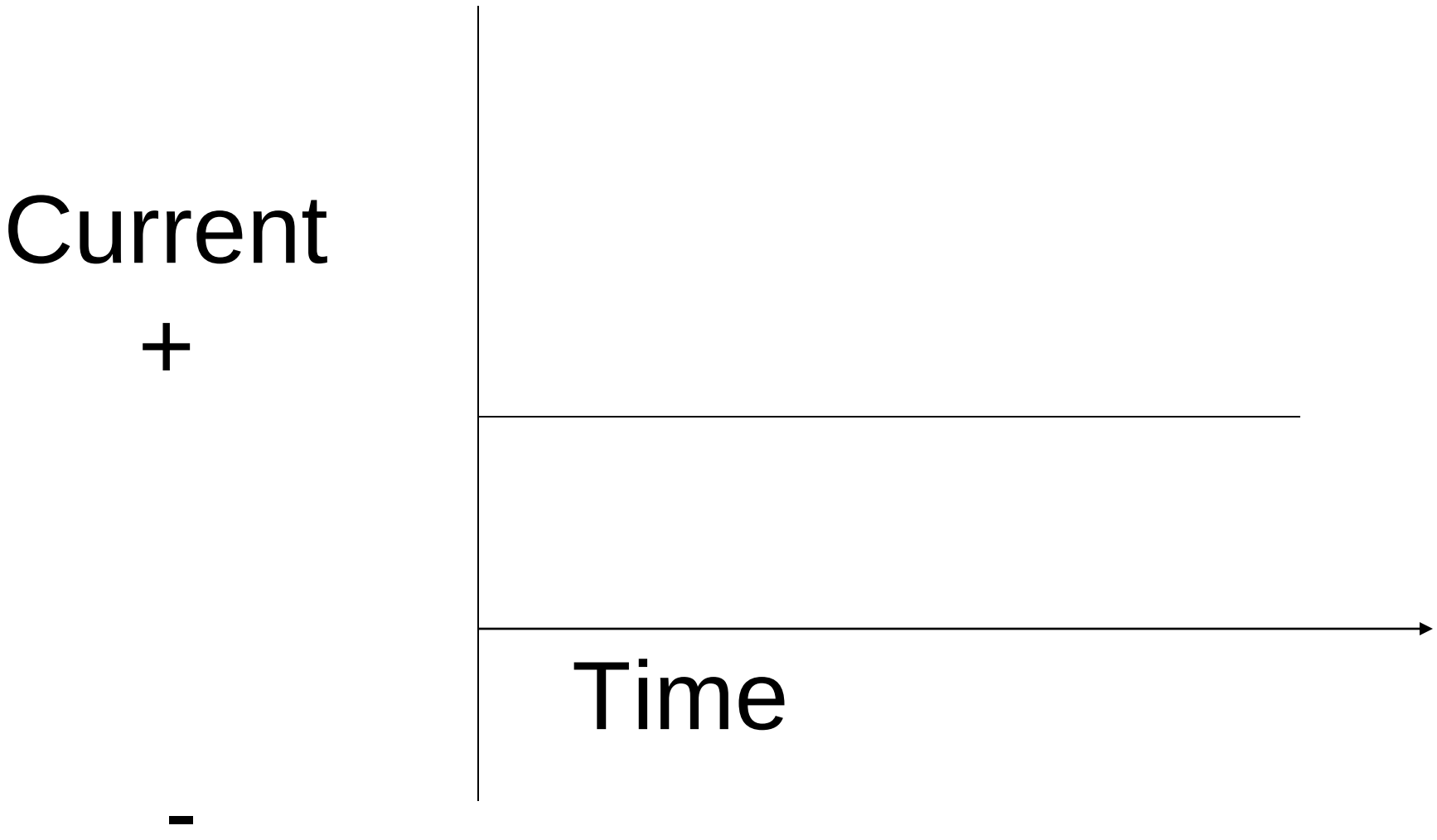
Solving for E in Ohm's law

$$E = IR$$

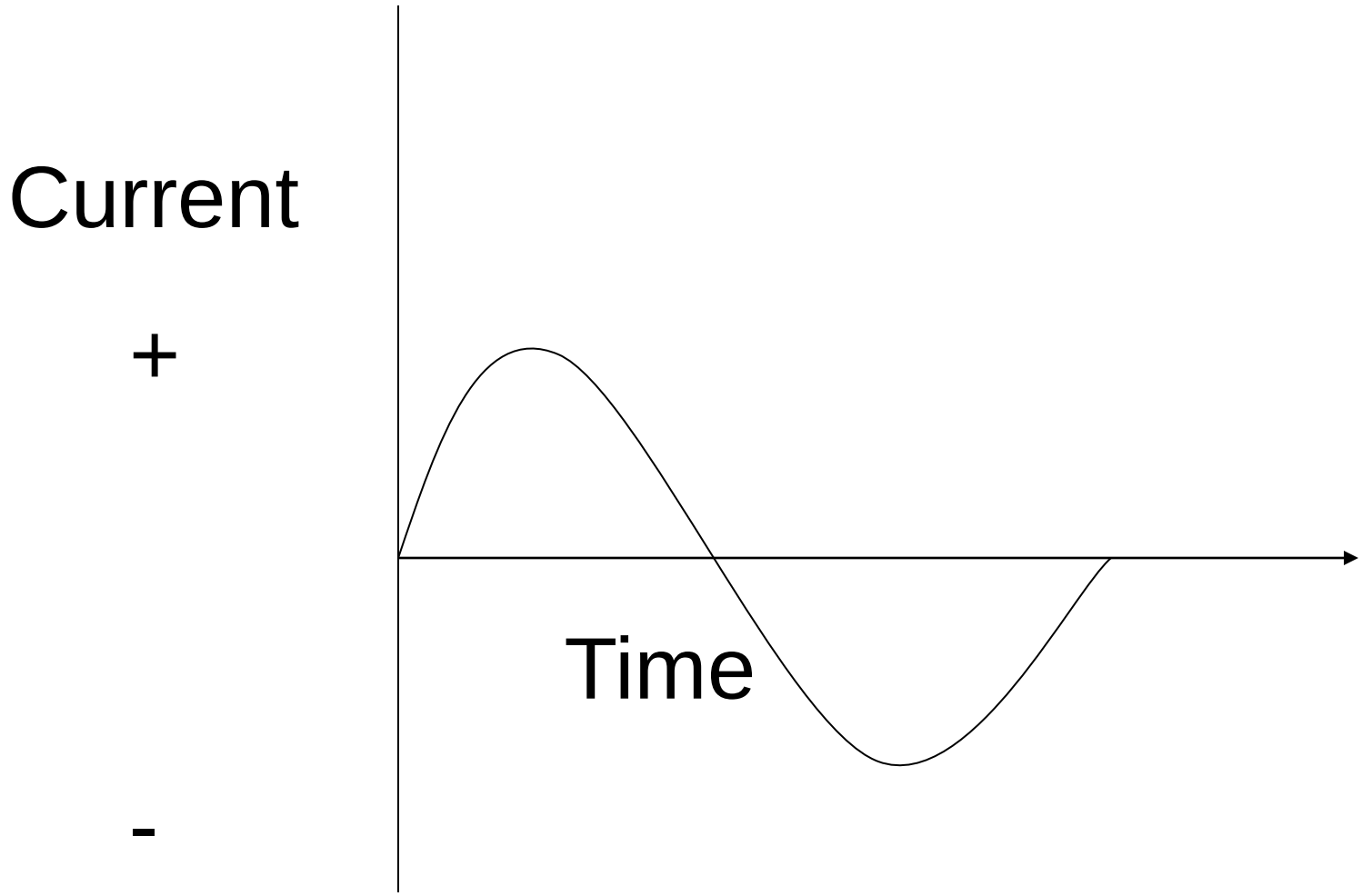
Important observation:

The voltage across a resistor (the *voltage drop*) is zero if I (the current) is zero.

Direct current (DC) flows in only one direction

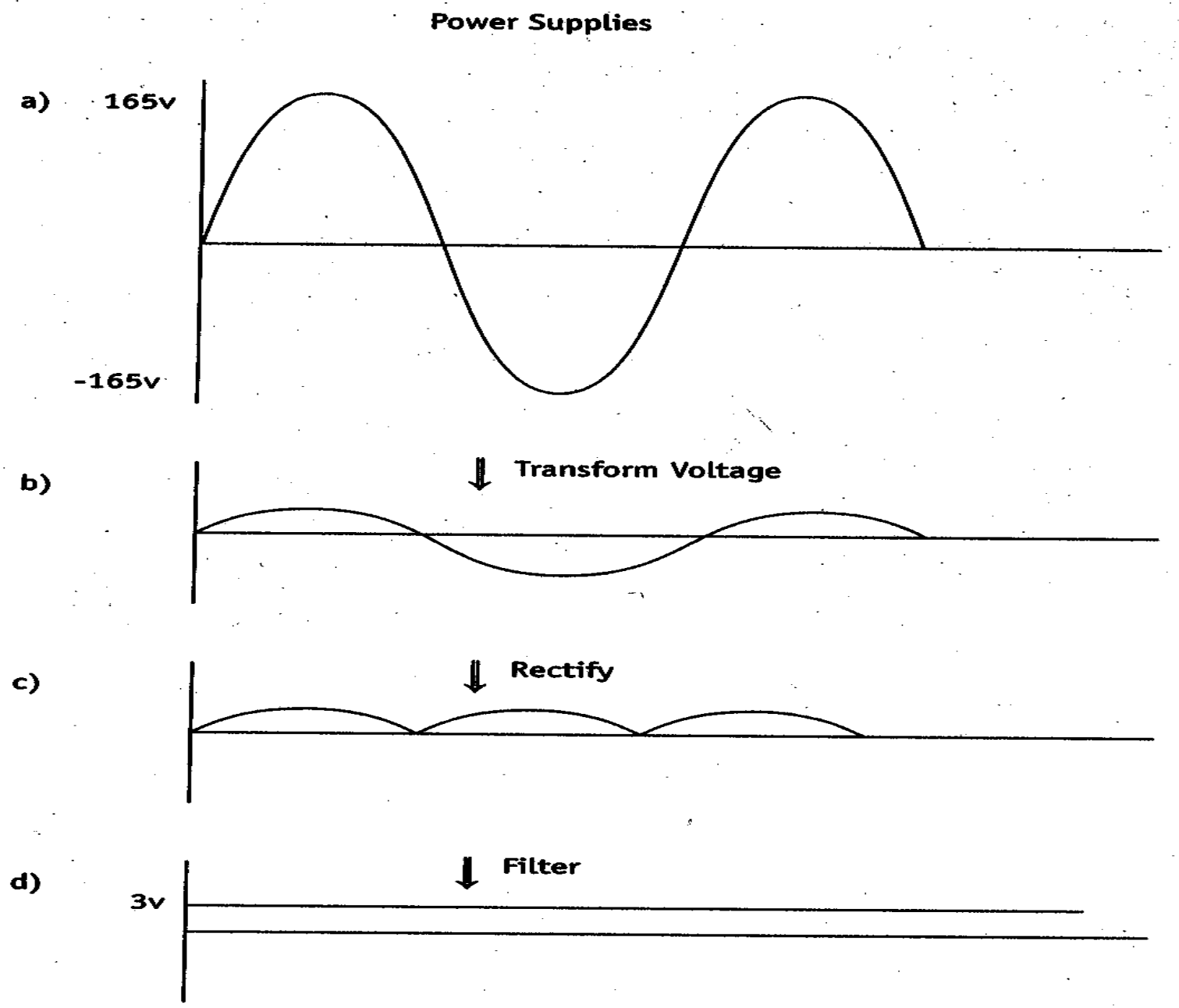


Alternating current (AC) repeatedly changes it direction of flow.



Computers circuits cannot use the 117 volt AC *line voltage* available by household electrical outlets. A computer *power supply* converts line voltage to the DC voltage required by computer circuits.

FIGURE 5.3


Power Supplies
a)
165v
-165v
b)
Transform Voltage
c)
Rectify
d)
Filter
3v

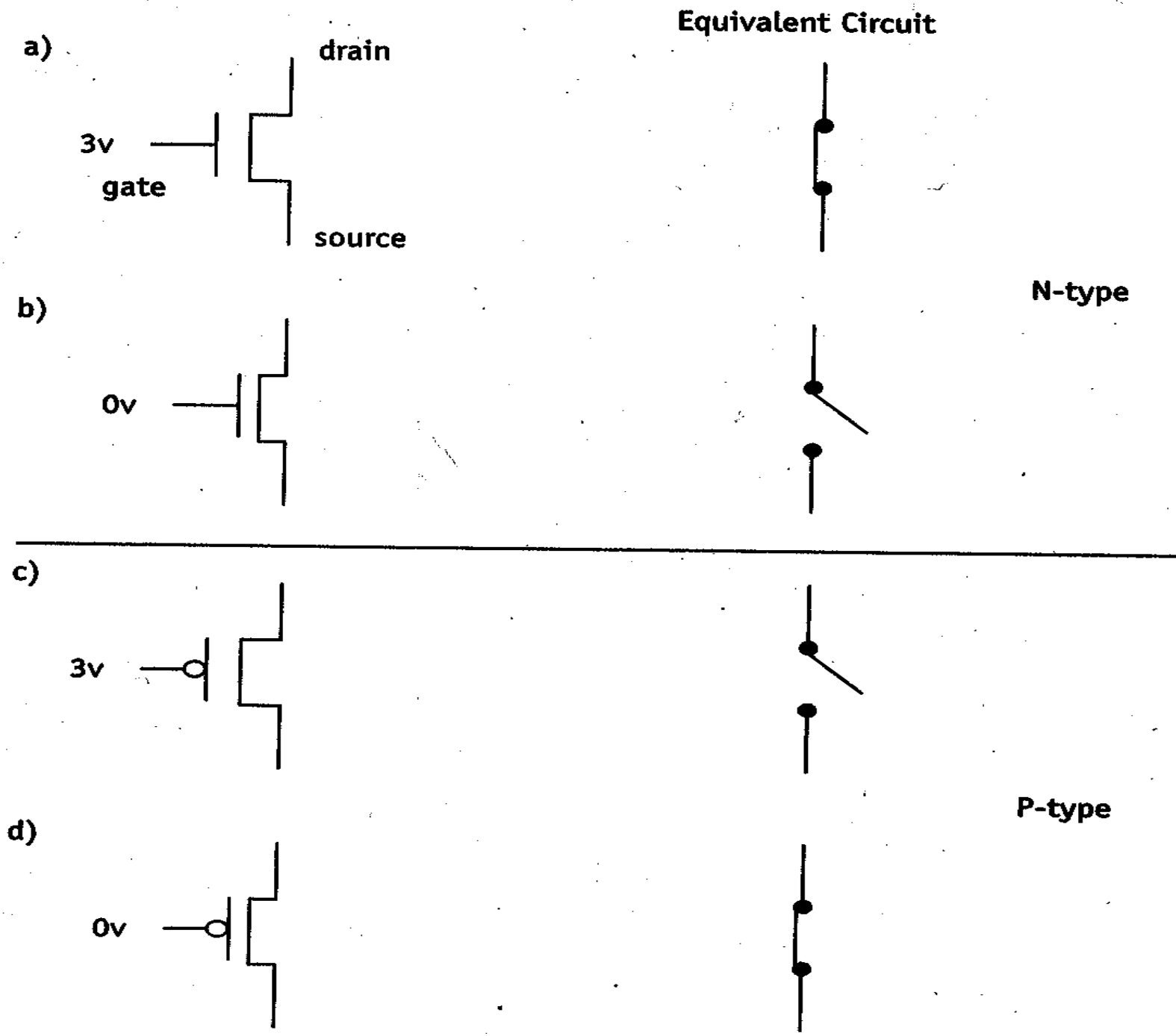
Safety rules with electricity

- Never touch any electrical equipment when any part of your body is wet.
- Never open electrical equipment unless you are qualified to do so.
- Keep one hand in your pocket while examining any electrical circuit.
- Never use any electrical equipment with frayed wires.

MOS transistors

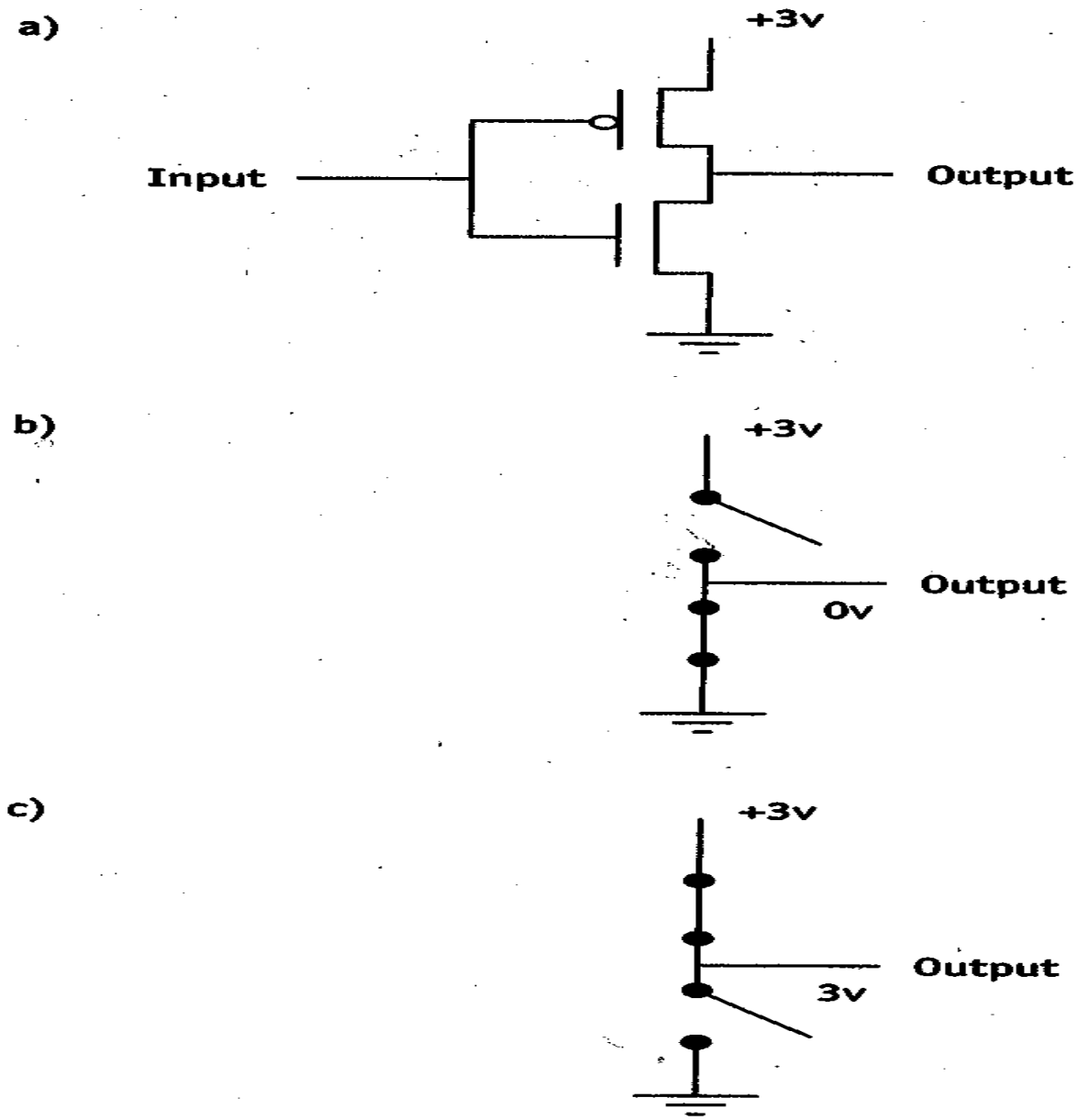
- Low power consumption
- High noise immunity
- High fan out
- Three leads: source, drain, gate
- Two types: PMOS (P-type), NMOS (N-type)
- CMOS technology: PMOS, NMOS used in pairs to minimize current requirements

FIGURE 5.4


Equivalent Circuit
a)
drain
3v
gate
source
b)
N-type
0v
c)
3v
d)
P-type
0v

Inverter (NOT gate)

FIGURE 5.5



Truth table for NOT

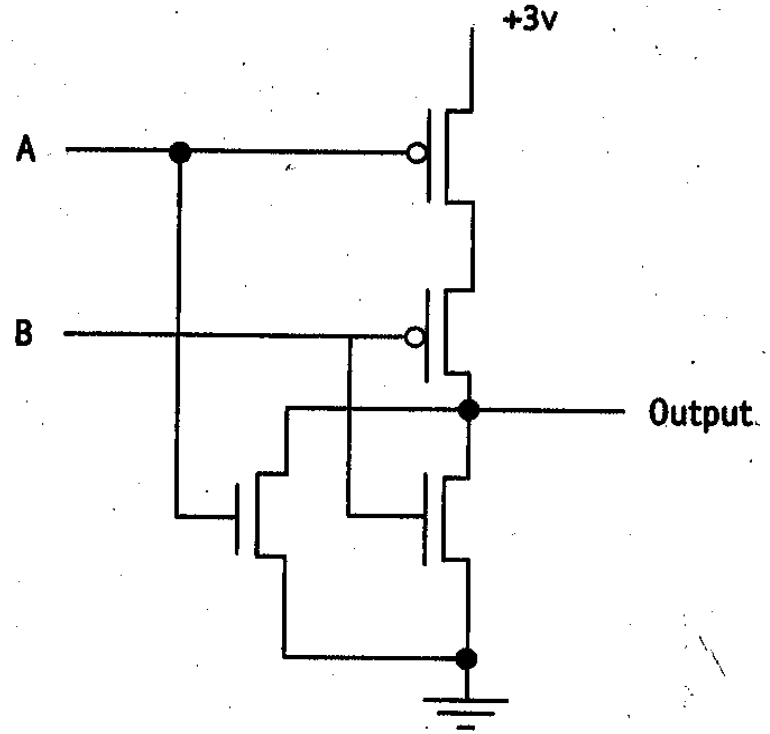
Input/output relationship:

in	out
0	3
3	1

If we let 0 and 3 volts represent the logic values 0 and 1, respectively, we get the truth table

in	out
0	1
1	0

FIGURE 5.6 a) NOR/NAND Function



b)

A	B	Output
0	0	3
0	3	0
3	0	0
3	3	0

c)

A	B	Output	
0	0	1	NOR
0	1	0	
1	0	0	
1	1	0	

Positive logic

d)

A	B	Output	
1	1	0	NAND
1	0	1	
0	1	1	
0	0	1	

Negative logic

Positive logic – the higher voltage value represents 1
Negative logic – the lower voltage value represents 1

A *capacitor* stores an electrical charge. When a capacitor is charging or discharging, current is flowing (and heat is generated).

FIGURE 5.7

Capacitance

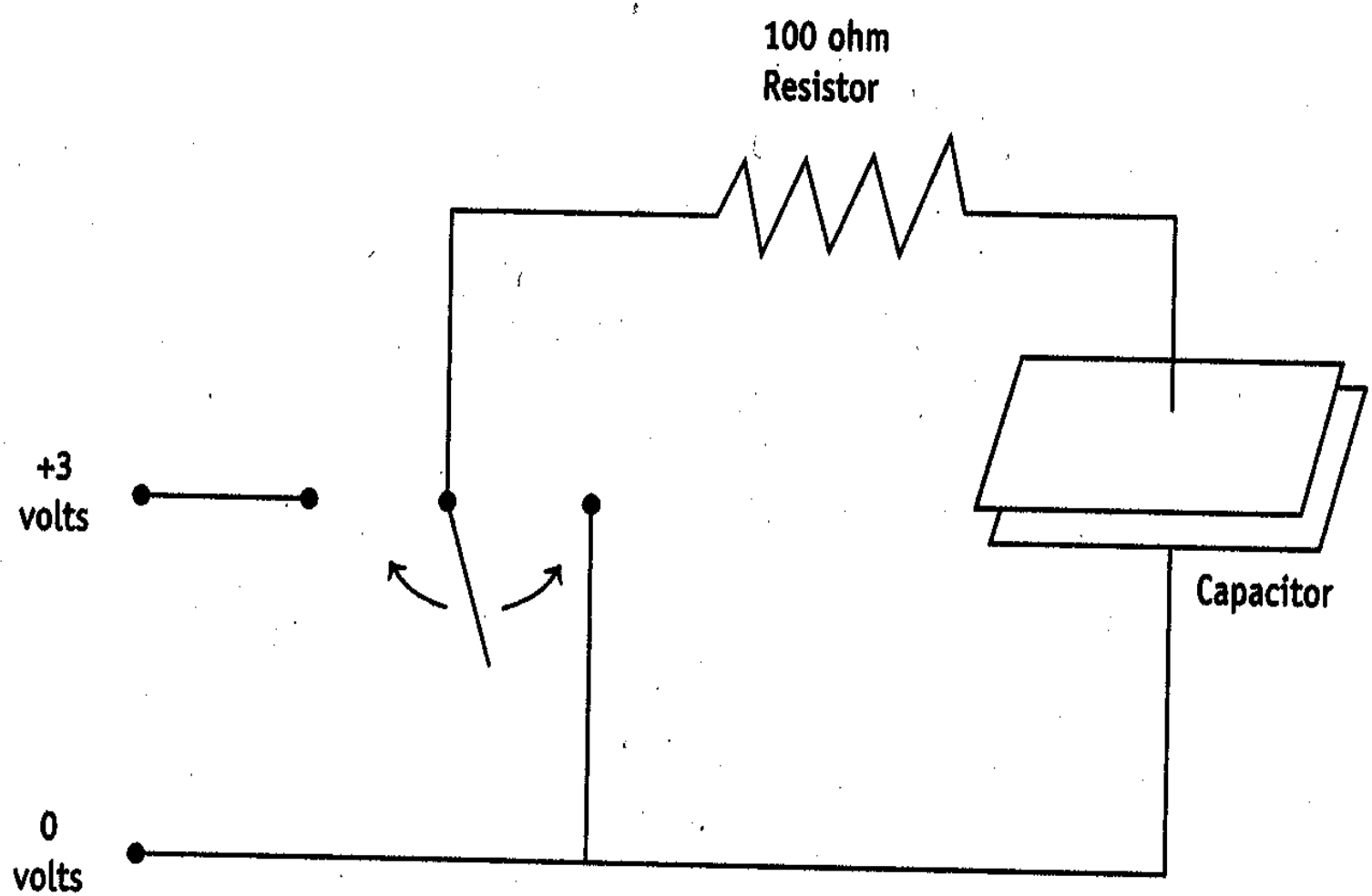
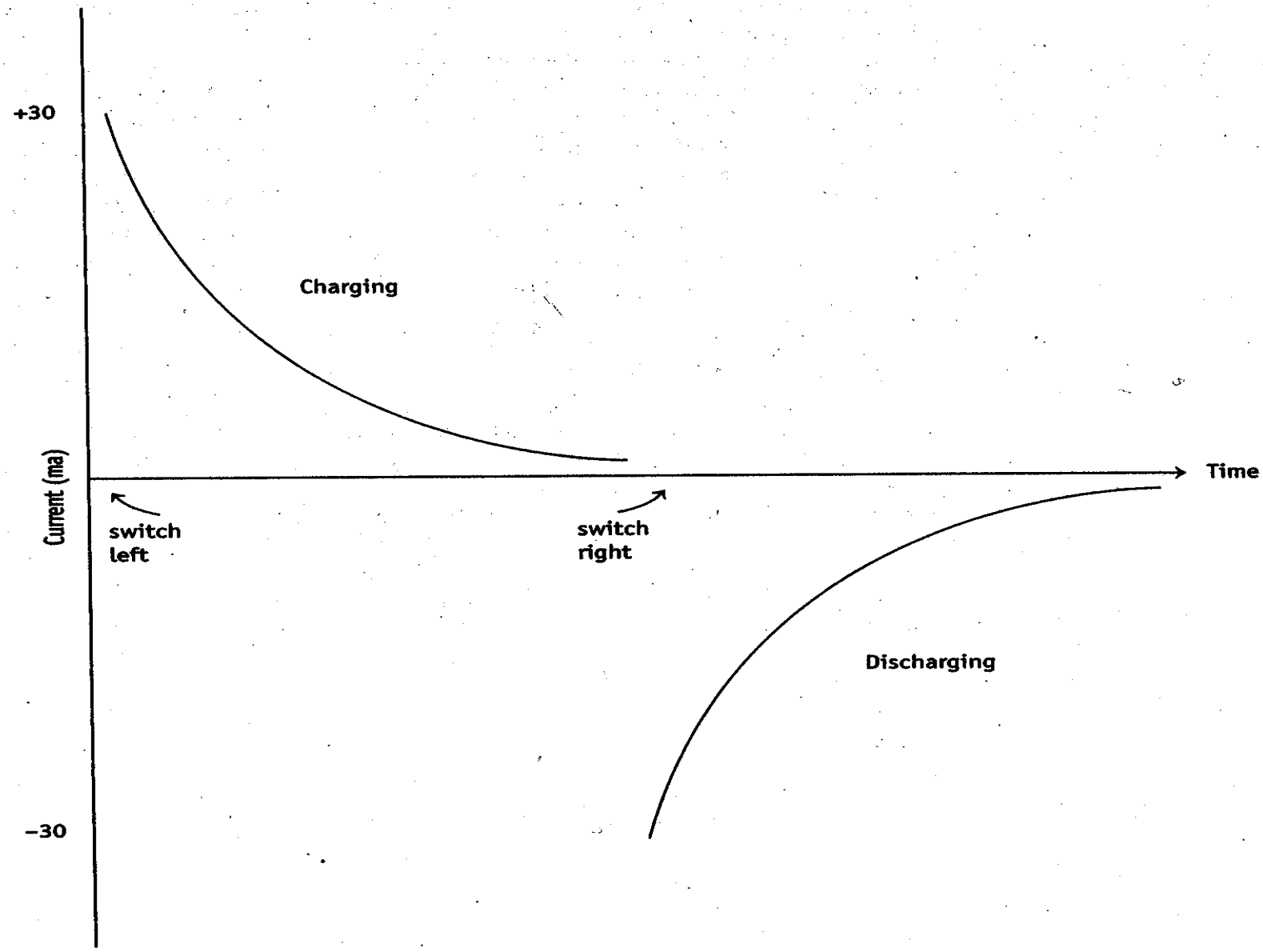


FIGURE 5.8 Resistor - Capacitor Circuit



The heat generated in our capacitor-resistor circuit is proportional to the frequency of voltage changes.

The heat generated is also proportional to the *square* of the magnitude of the voltage change.

Problem:

All circuits, including computer circuits, have capacitance.

Increasing voltage or frequency produces more heat.

Computer circuits can be damaged by excessive heat.

Solutions:

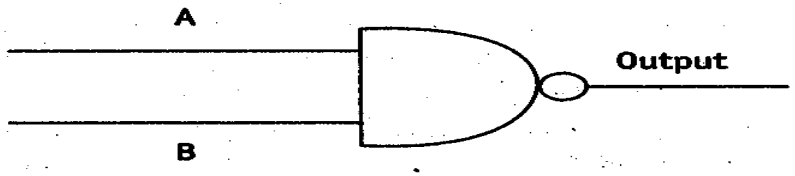
Reduce voltage to compensate for increasing frequency.
But must not make voltage separation too small.

Attach cooling devices.

Combinational Circuits

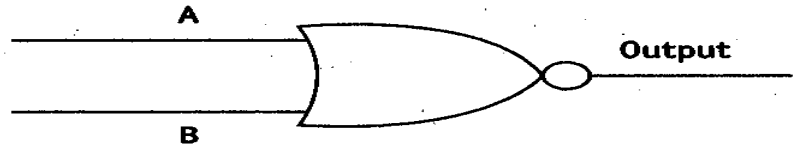
FIGURE 5.9

NAND



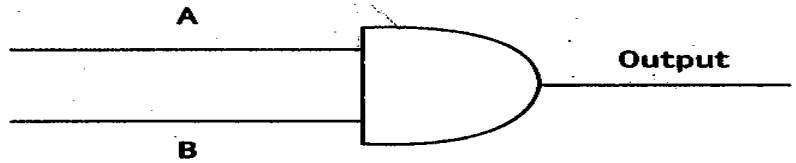
A	B	Output
0	0	1
0	1	1
1	0	1
1	1	0

NOR



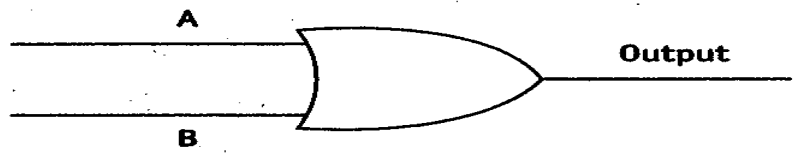
A	B	Output
0	0	1
0	1	0
1	0	0
1	1	0

AND



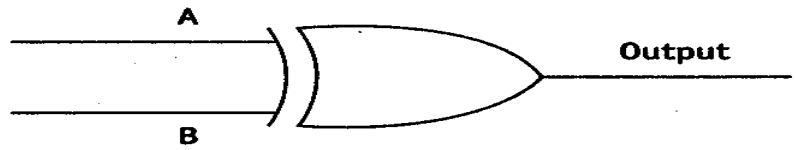
A	B	Output
0	0	0
0	1	0
1	0	0
1	1	1

OR



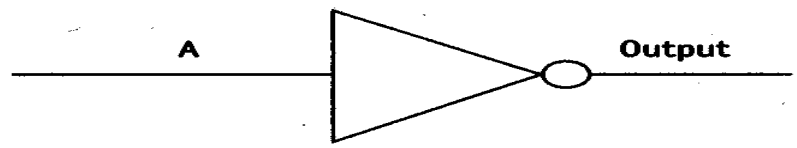
A	B	Output
0	0	0
0	1	1
1	0	1
1	1	1

XOR



A	B	Output
0	0	0
0	1	1
1	0	1
1	1	0

NOT



A	Output
0	1
1	0

Building AND and OR gates

FIGURE 5.10

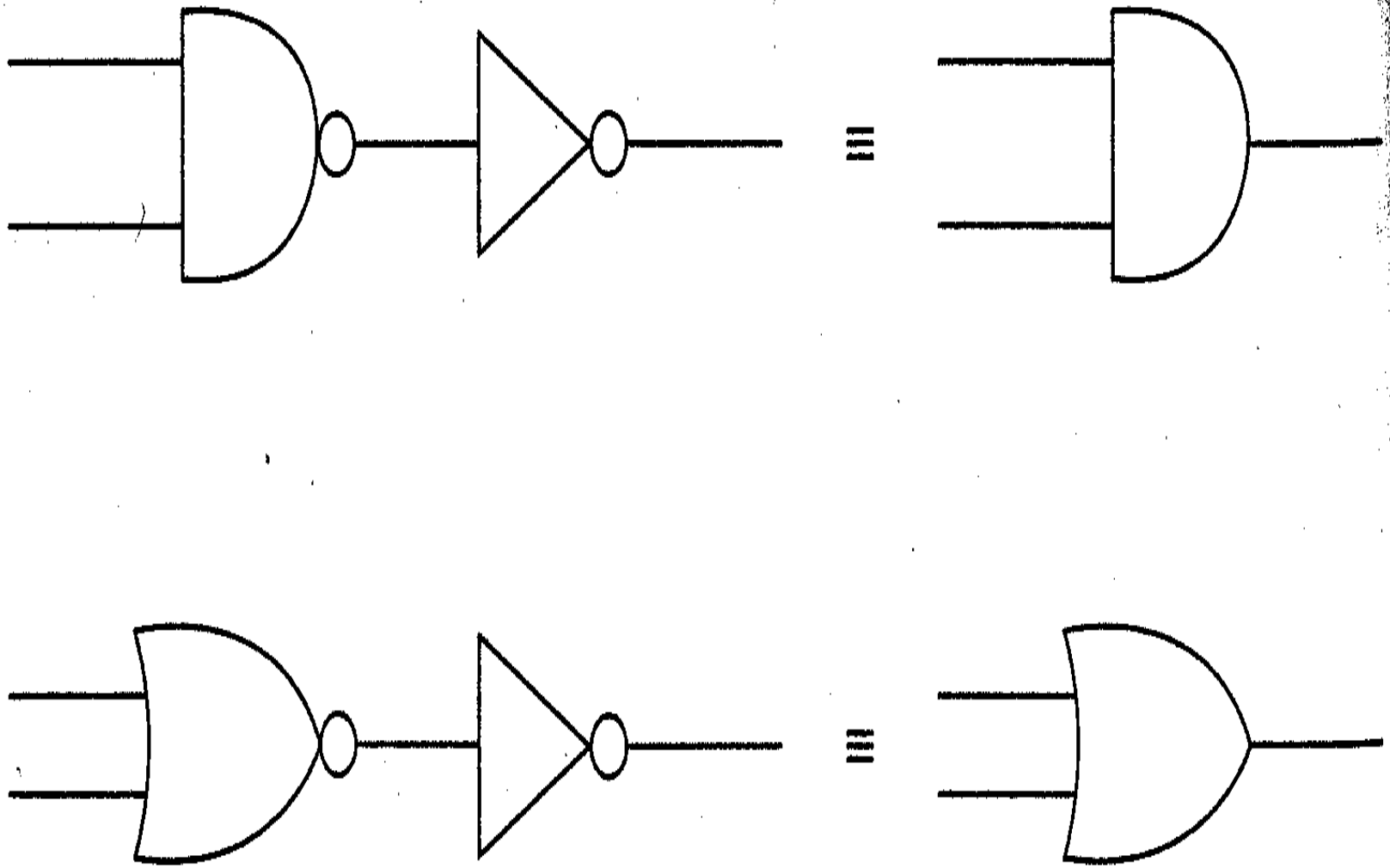
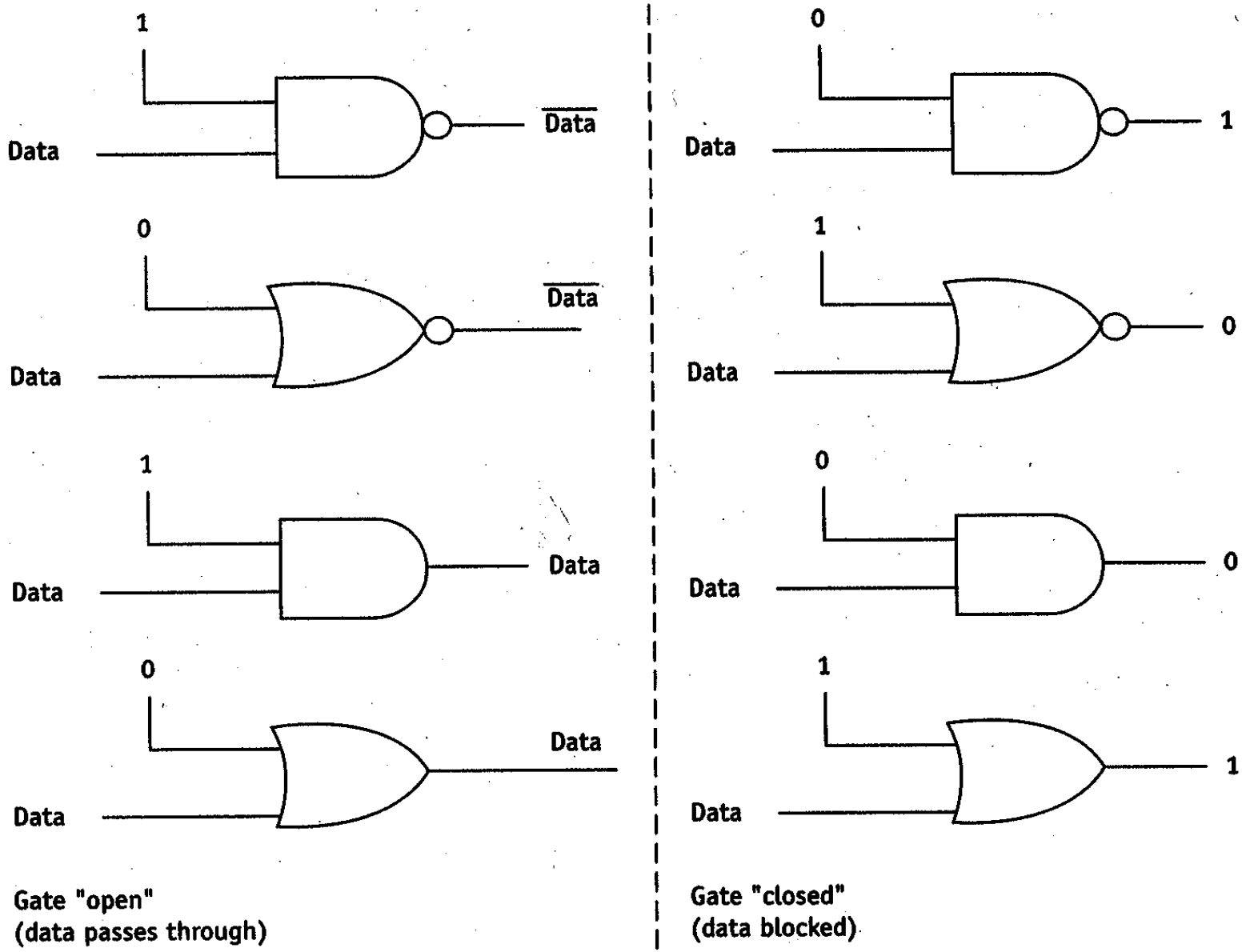


FIGURE 5.11 Gating Action



It is easy to construct the circuit corresponding to any truth table.

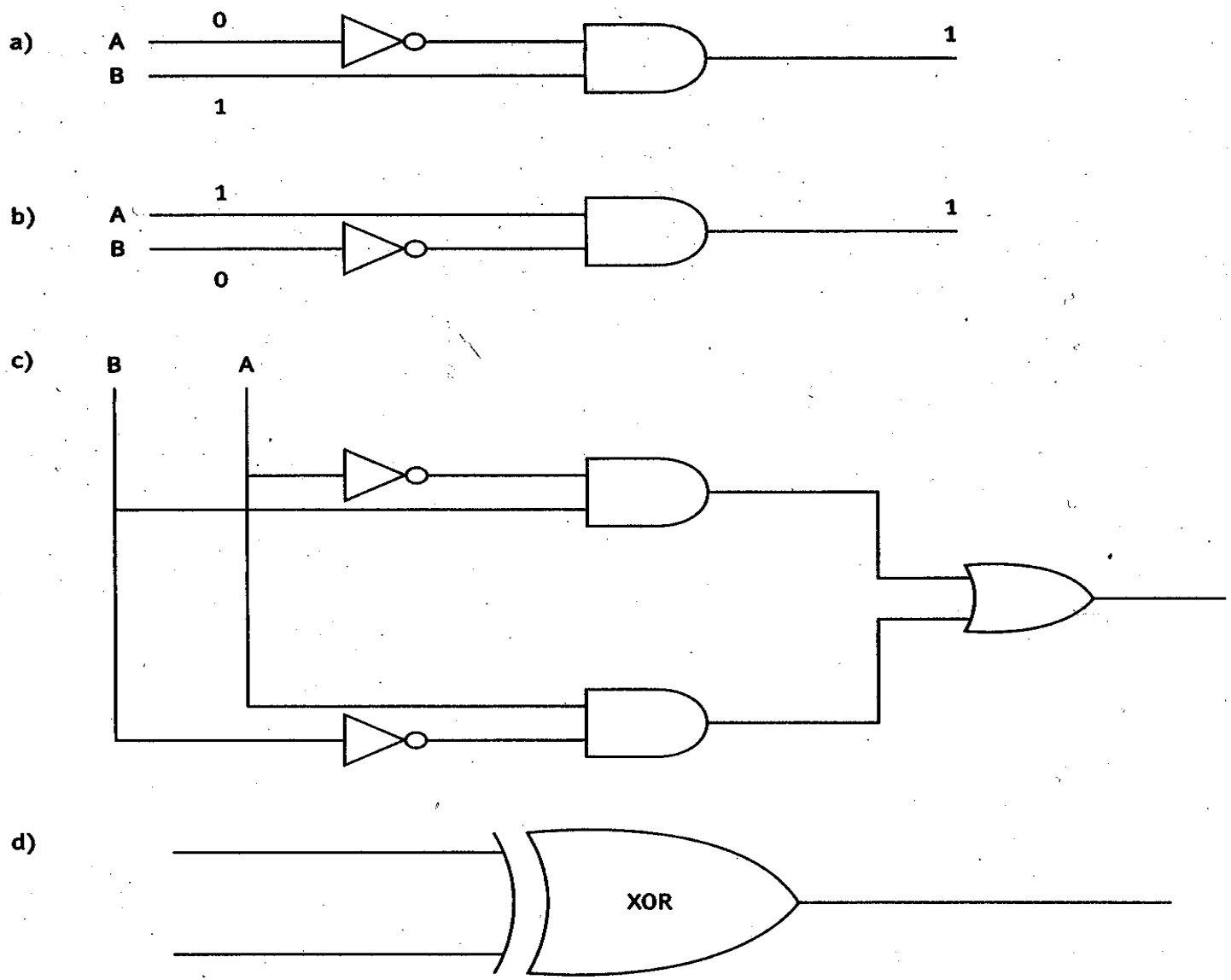
Implementing circuit from truth table

FIGURE 5.12

A	B	C	
0	0	0	
0	1	1	← select this row
1	0	1	← select this row
1	1	0	

Build a circuit for each row with a 1 output, and then connect using an OR gate.

FIGURE 5.13 Implementing a Function



Note the resulting circuit contains AND gates driving an OR gate.

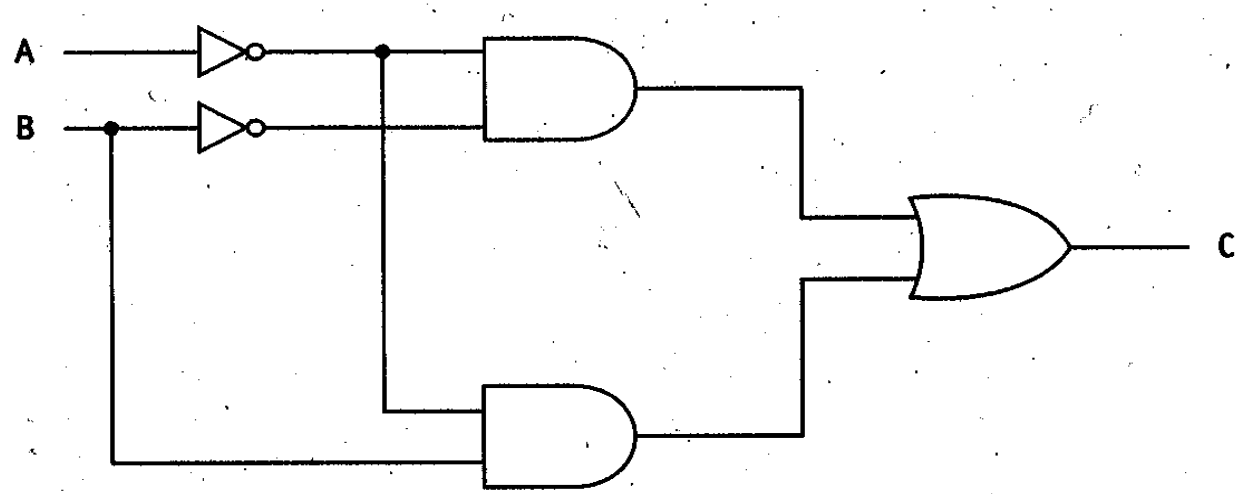
There is more than one way to implement a truth table. The circuit with the least number of gates is called the *minimal* circuit.

FIGURE 5.14 Minimization

a)

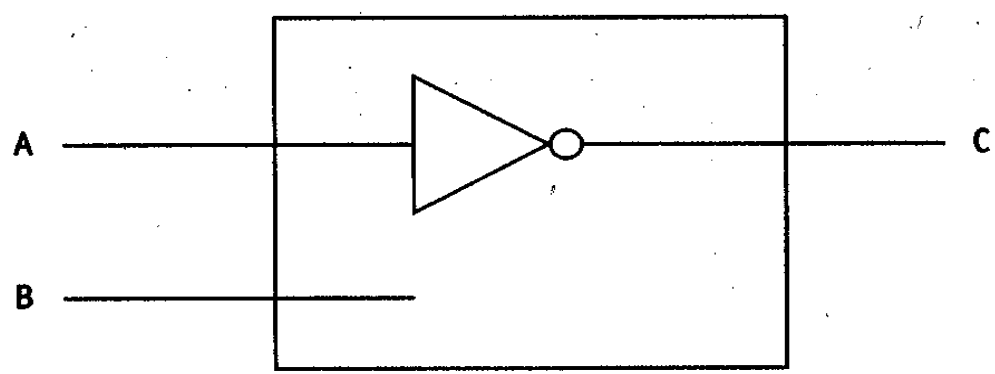
A	B	C
0	0	1
0	1	1
1	0	0
1	1	0

b)



Not Minimized

c)



Minimized

Circuits can be minimized using any of several techniques. We will study two such techniques: Boolean algebra and Karnaugh maps.

Boolean expressions

- Describe operation of computer circuits
- ‘+’ represents OR
- Concatenation represents AND
- Superscript bar represents NOT
- Two values: 1 and 0

$$C = \overline{A}\,\overline{B} + \overline{A}\,B$$

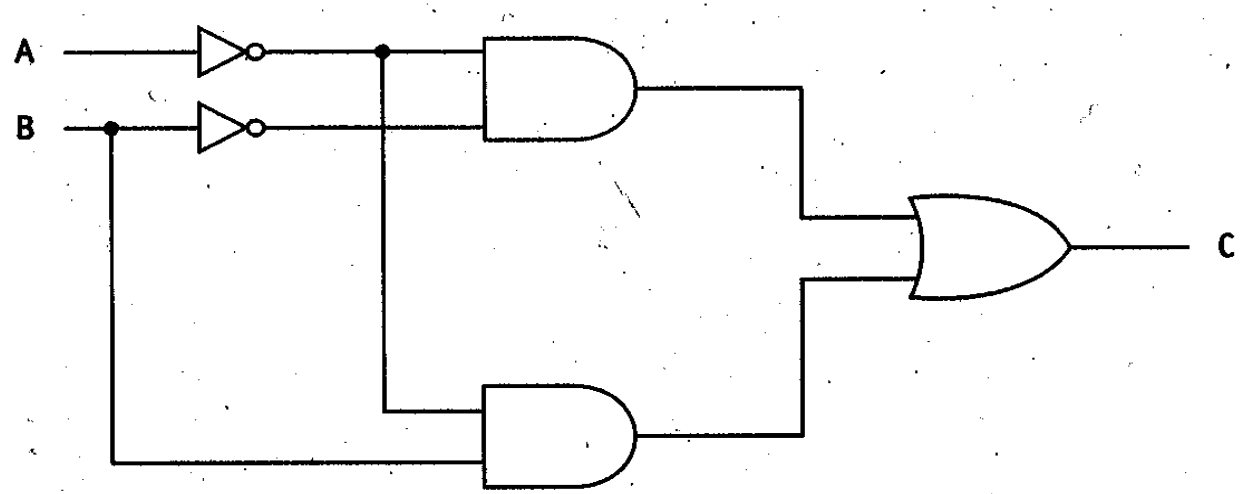
Describes of circuit on the next slide.

FIGURE 5.14 Minimization

a)

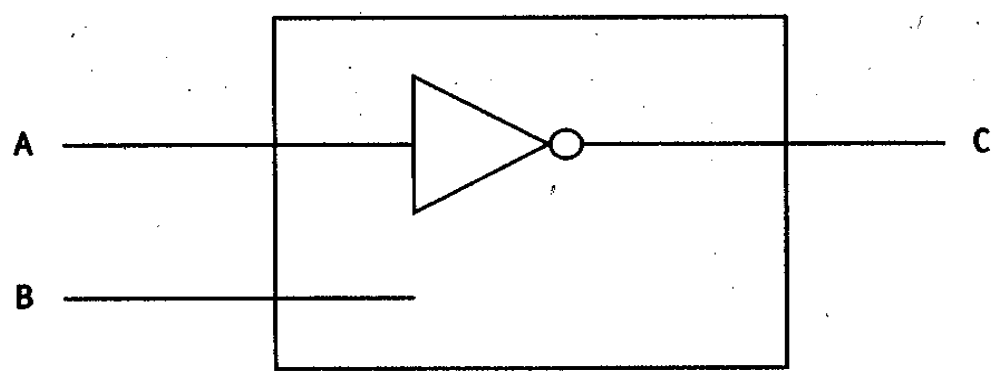
A	B	C
0	0	1
0	1	1
1	0	0
1	1	0

b)



Not Minimized

c)



Minimized

You can use Boolean algebra to simplify this expression (hint: factor out “A bar”).

$$C = \overline{A}\,\overline{B} + \overline{A}\,B$$

Using these laws, we can manipulate Boolean expressions into equivalent simpler expressions. For example, let's use Boolean algebra to simplify

$$C = \overline{A}\,\overline{B} + \overline{A}\,B$$

We get

$$\begin{aligned} C &= \overline{A}\,\overline{B} + \overline{A}\,B \\ &= \overline{A}(\overline{B} + B) && \text{by the distributive law} \\ &= \overline{A}(1) && \text{by the inverse law} \\ &= \overline{A} && \text{by the identity law} \end{aligned}$$

The laws of Boolean algebra are not identical to the laws of real number algebra.

FIGURE 5.15

Distributive Law	$A(B + C) = AB + AC$
	$A + BC = (A + B)(A + C)$
Commutative Law	$AB = BA$
	$A + B = B + A$
Absorption Law	$A + AB = A$
	$A(A + B) = A$
Identity Law	$A1 = A$
	$A + 0 = A$
Null Law	$A0 = 0$
	$A + 1 = 1$
Idempotent Law	$A + A = A$
	$AA = A$
Inverse Law	$A\overline{A} = 0$
	$A + \overline{A} = 1$
Associative Law	$(A + B) + C = A + (B + C)$
	$(AB)C = A(BC)$
DeMorgan's Laws	$\overline{A + B} = \overline{A}\,\overline{B}$
	$\overline{AB} = \overline{A} + \overline{B}$

Bye, bye black sheep rule:
If two terms differ in only one variable, may combine terms by eliminating “black sheep” variable.

$$ABCD + AB\overline{C}D$$

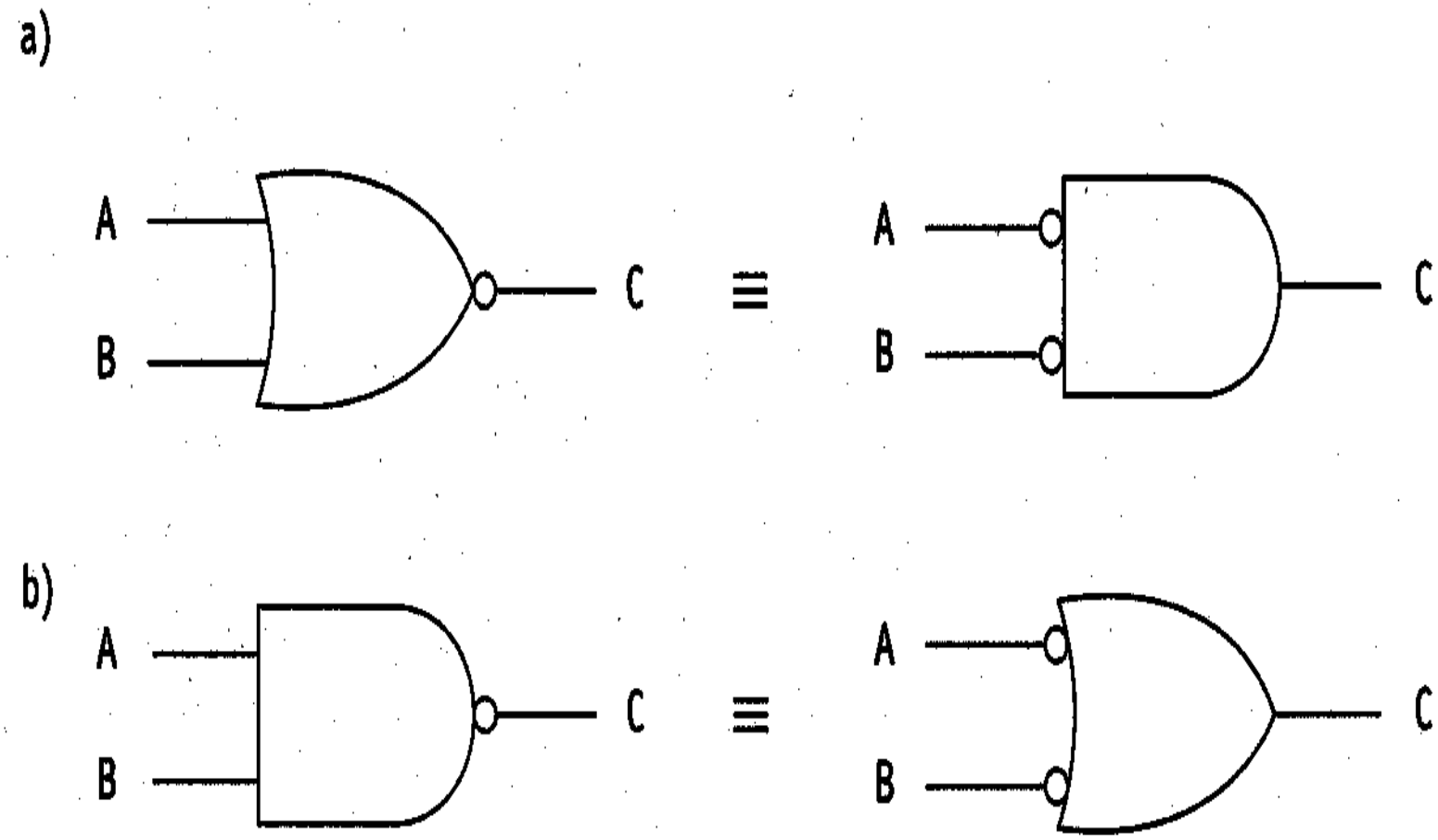
Black sheep rules yields single term **ABD**

DeMorgan's Laws

$$\overline{A + B} = \overline{A}\,\overline{B}$$

$$\overline{AB} = \overline{A} + \overline{B}$$

FIGURE 5.16 De Morgan's Laws

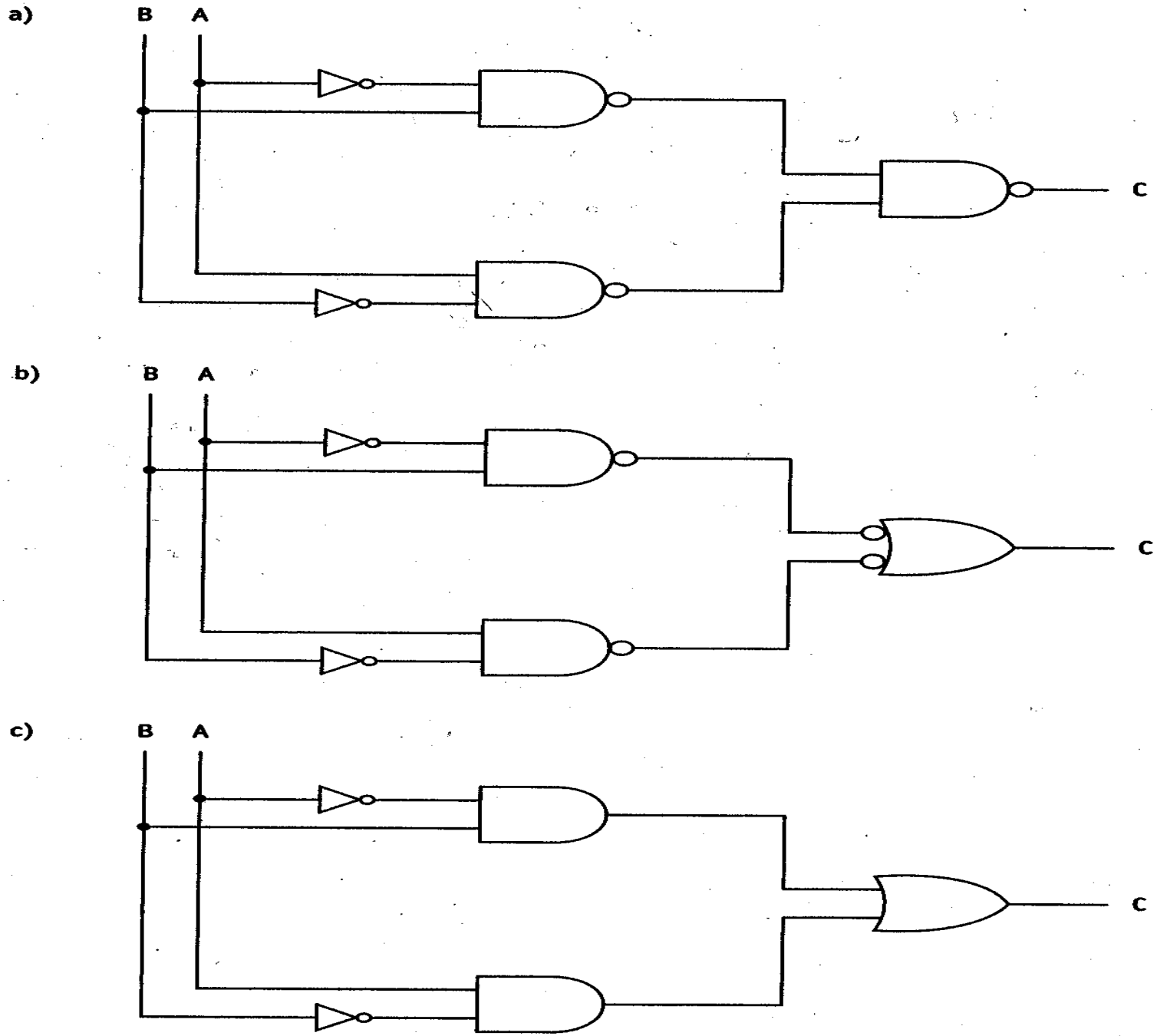


Move bubble back: gate changes

In a circuit consisting of AND gates driving a single OR gate, all AND and OR gates can be replaced by NAND gates. The resulting circuit will realize the same function.

FIGURE 5.17

NAND Gate Substitution

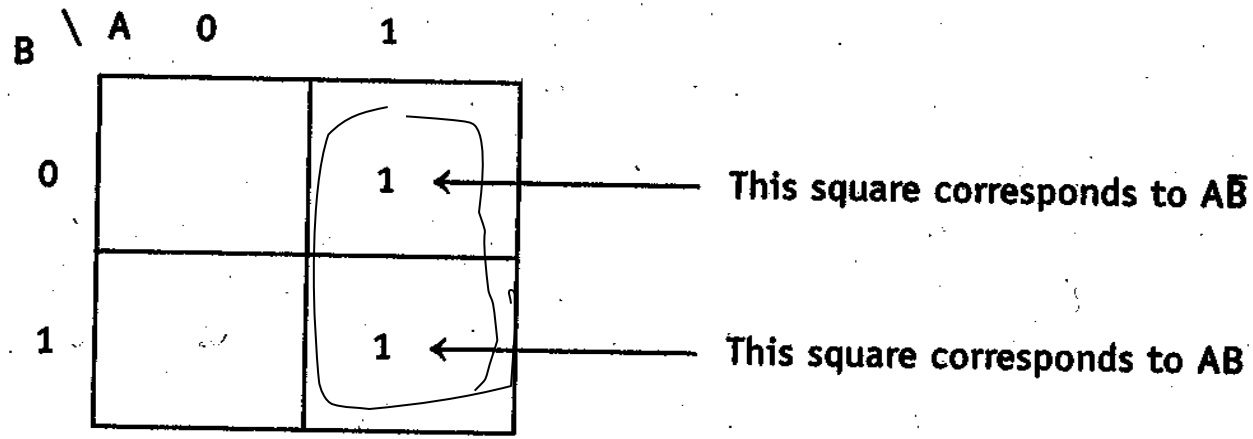


Karnaugh map

- Matrix representation of a Boolean function
- Useful for circuit minimization
- Horizontally and vertically adjacent squares differ in only one variable. Thus, horizontal and vertical groups correspond to terms simplified by the black sheep rule.

FIGURE 5.18

a)



b)

A	B	C
0	0	0
0	1	0
1	0	1
1	1	1

$$C = A\overline{B} + AB$$

Black sheep rule yields

$$C = A$$

When the function below is simplified, you have to use AB twice.

$$
\begin{aligned}
C &= A\overline{B} + AB + \overline{A}B \\
&= A\overline{B} + AB + AB + \overline{A}B && \text{by the idempotent law} \\
&= (A\overline{B} + AB) + (AB + \overline{A}B) && \text{by the associative law} \\
&= A + (AB + \overline{A}B) && \text{by the black sheep rule} \\
&= A + B && \text{by the black sheep rule}
\end{aligned}
$$

Now simplify same function using Karnaugh map. Note: using bottom right term (AB) twice

FIGURE 5.19

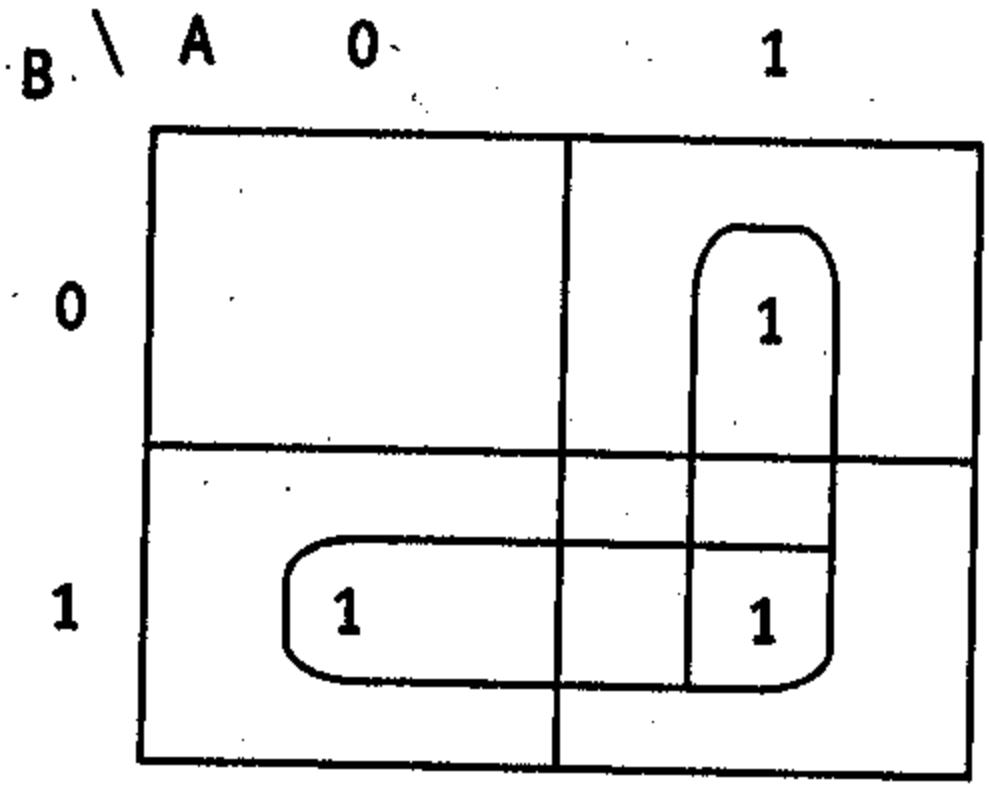
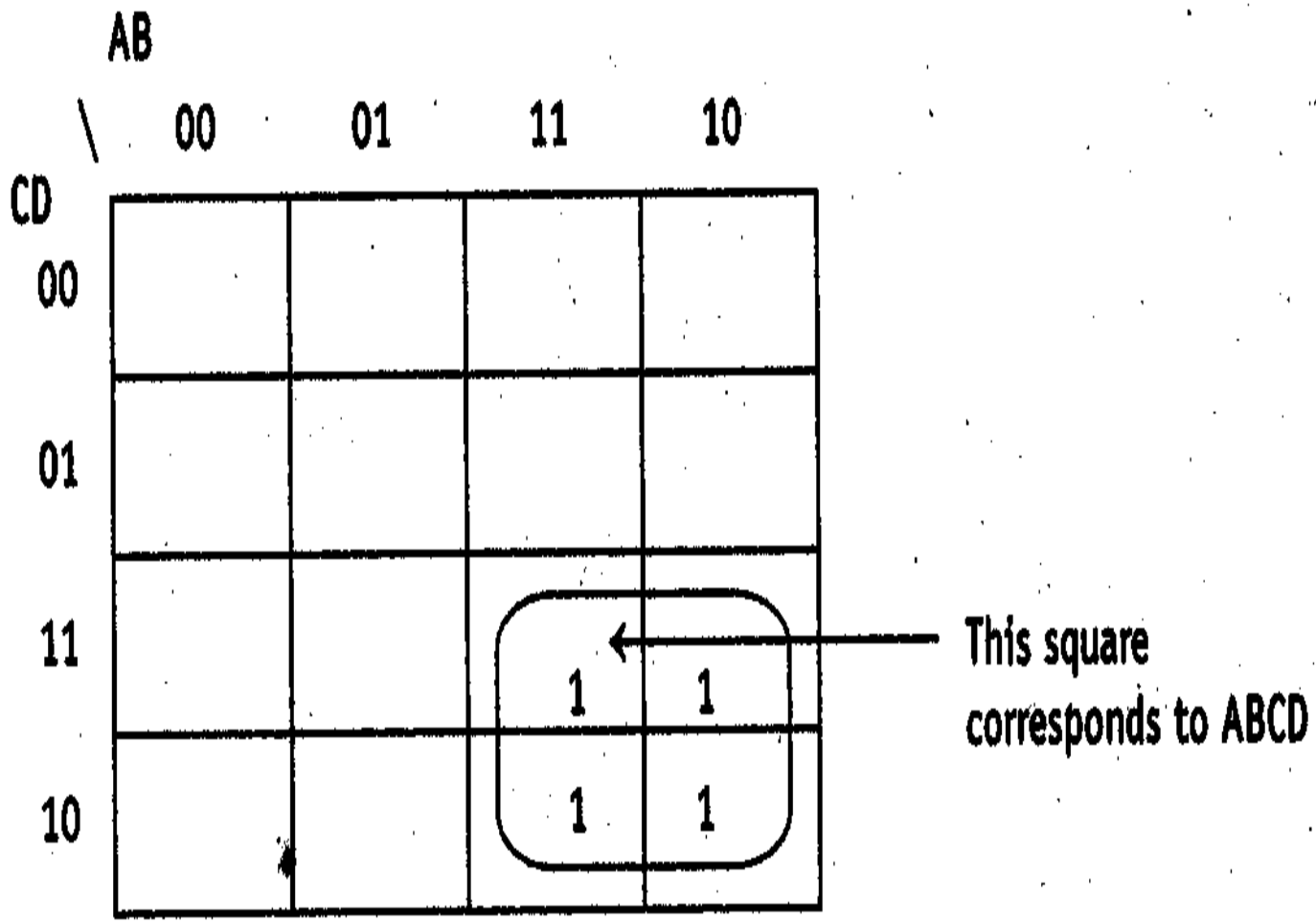
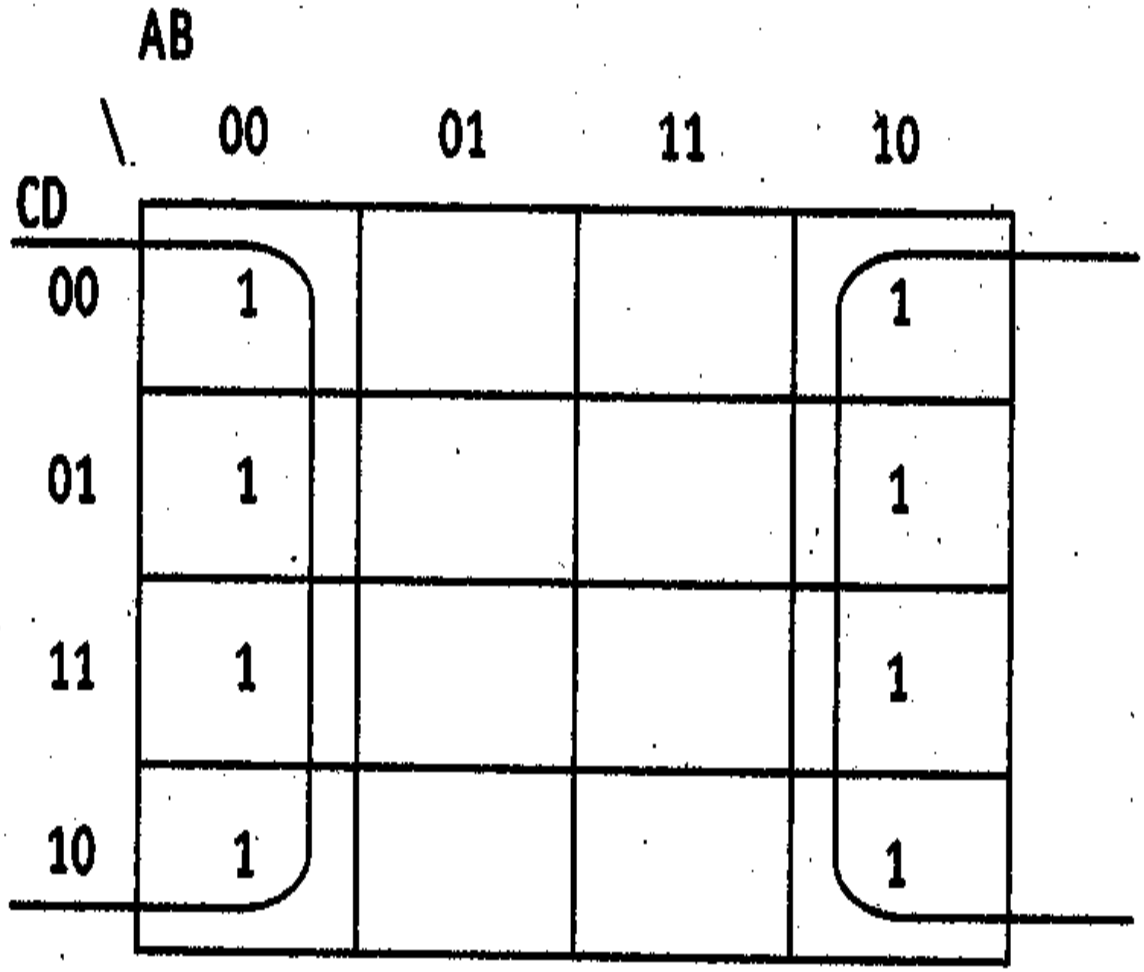


FIGURE 5.20



Group corresponds to AC

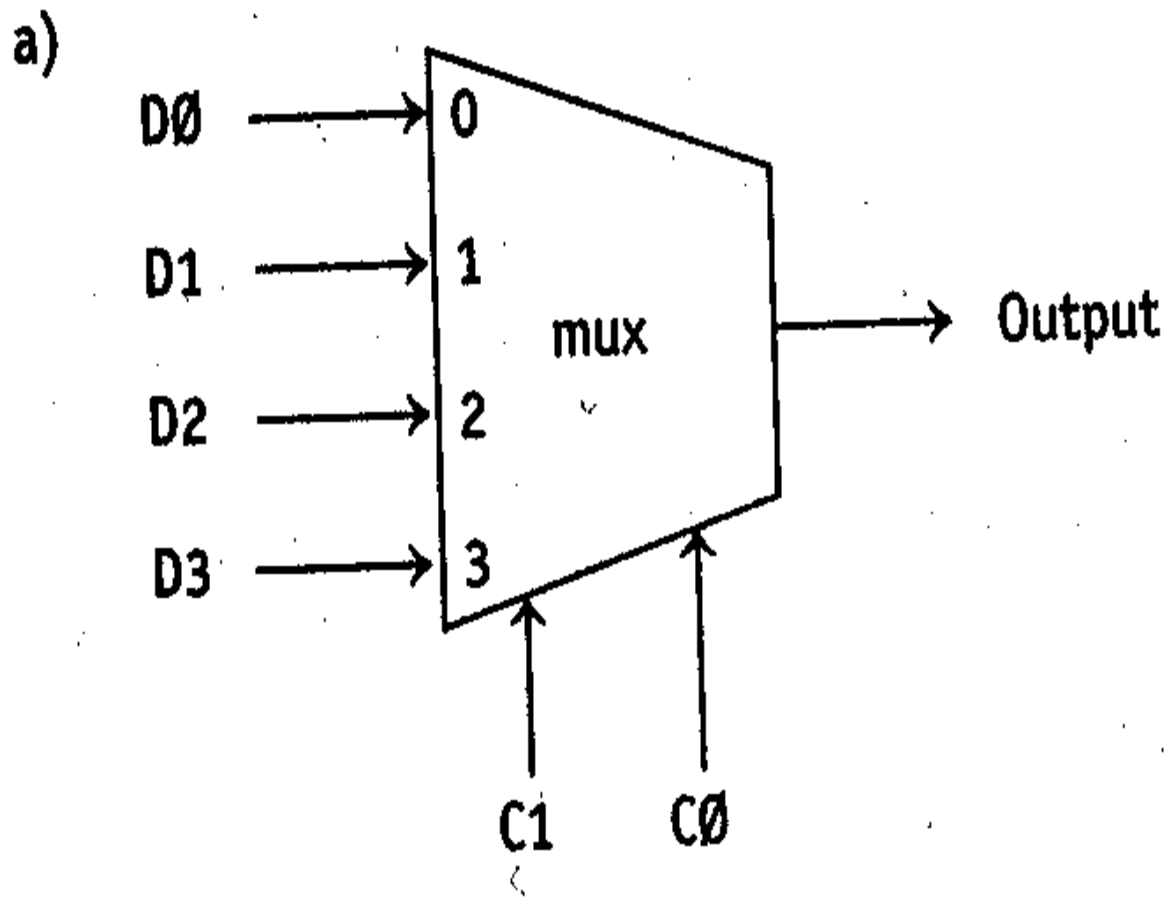
FIGURE 5.21



Group corresponds to $\overline{B}$

We now will use our basic gates to build more complex circuits

FIGURE 5.22 4-Input Multiplexer



Implementation of MUX

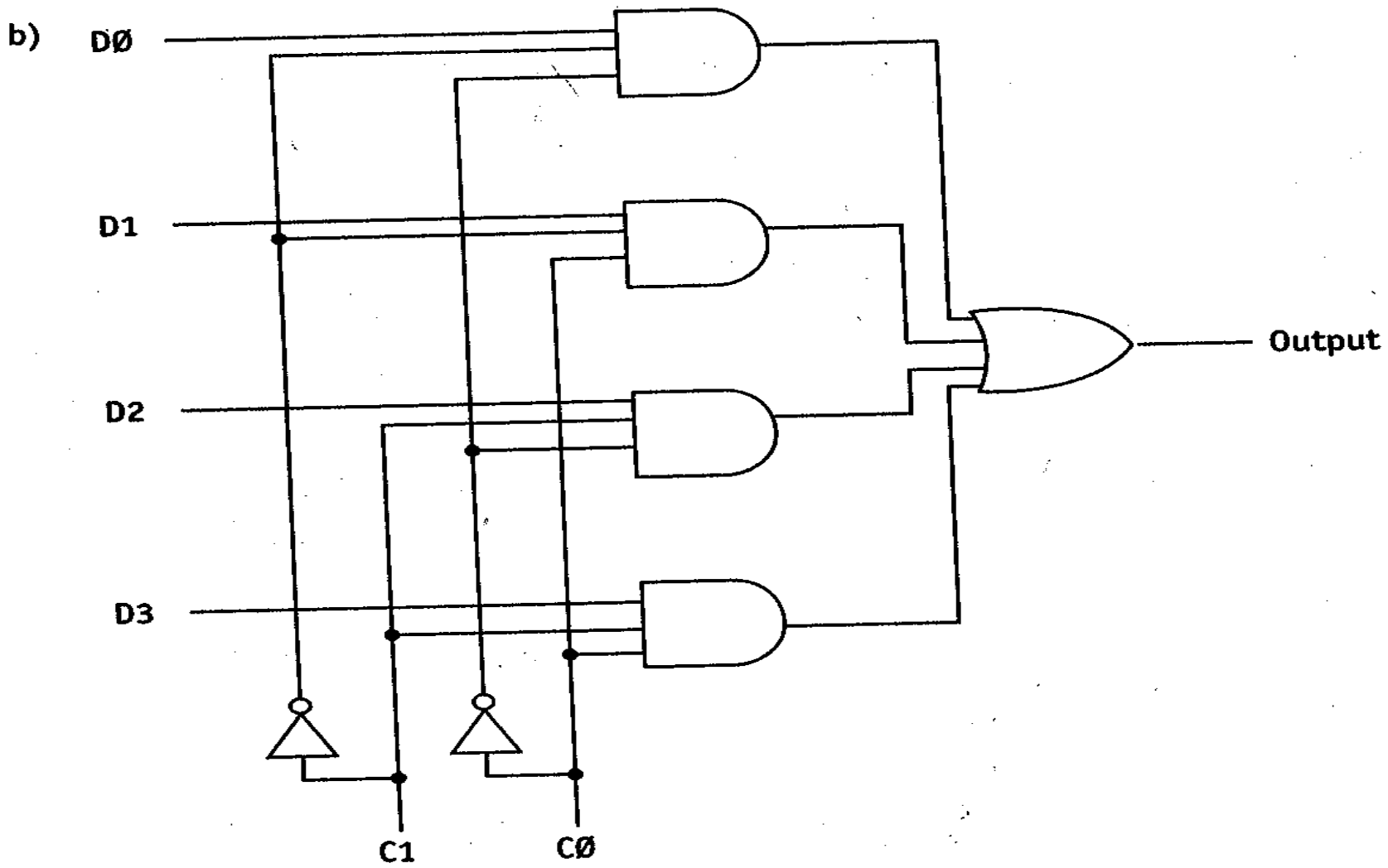
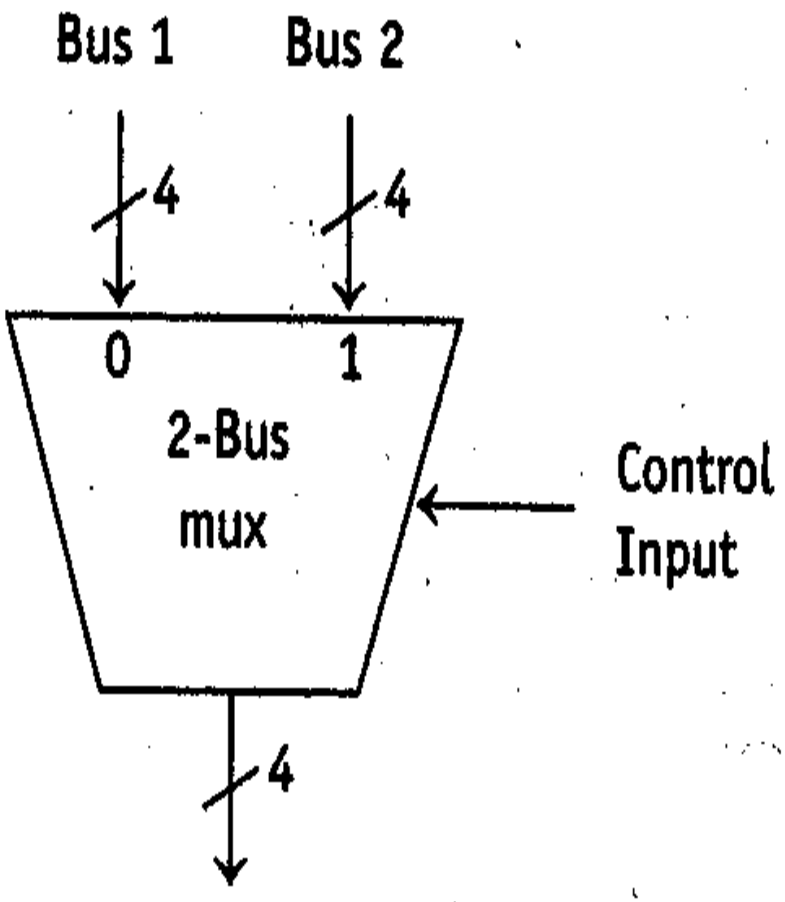


FIGURE 5.23

Two-Bus Input Multiplexer

a)



Implementation of 2 bus MUX

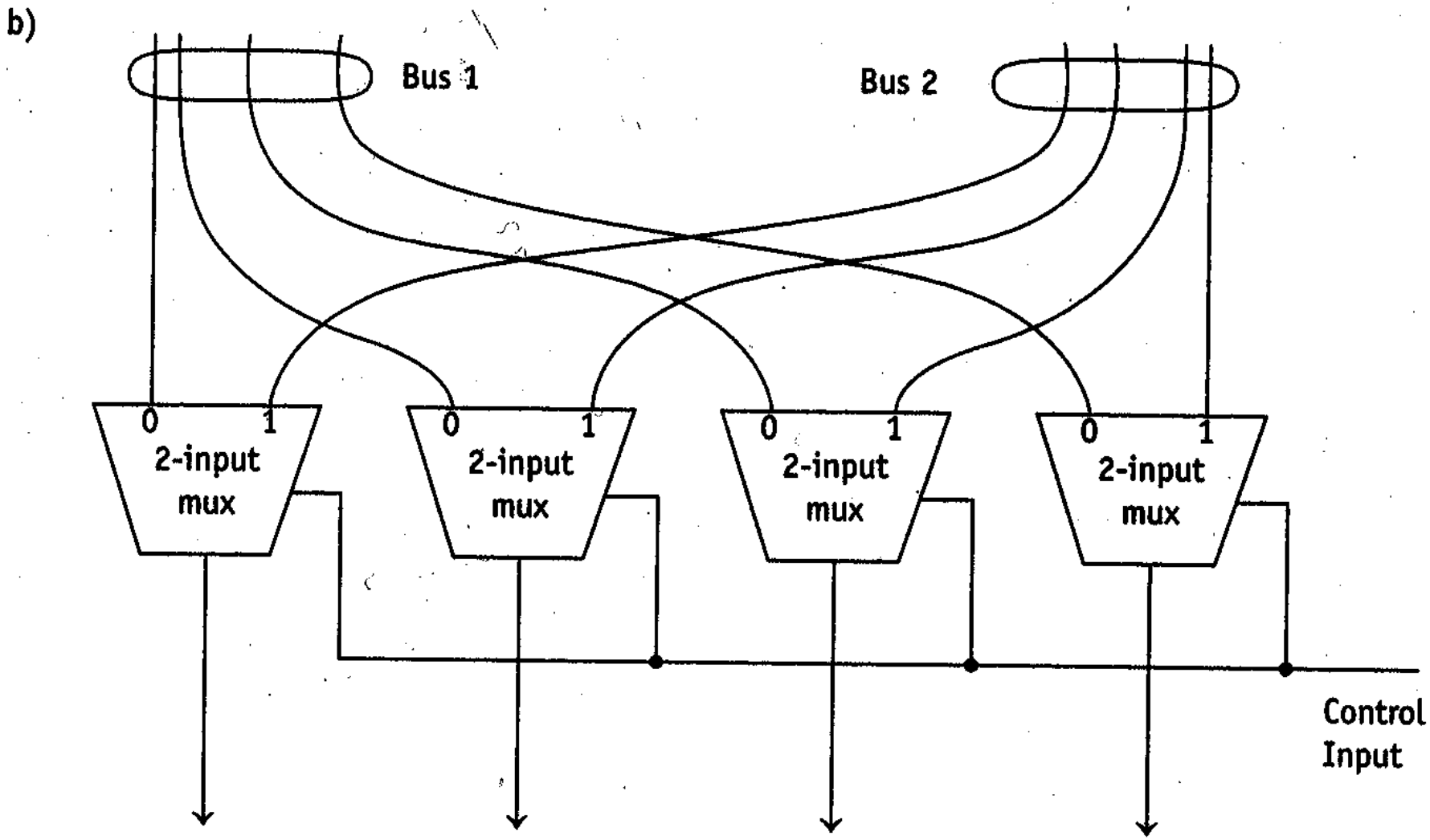
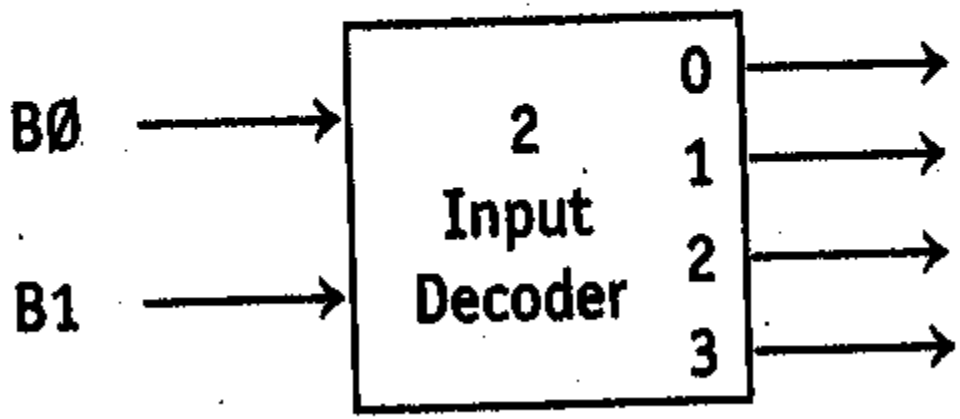


FIGURE 5.24 Two-Input Decoder



Implementation of 2 input decoder

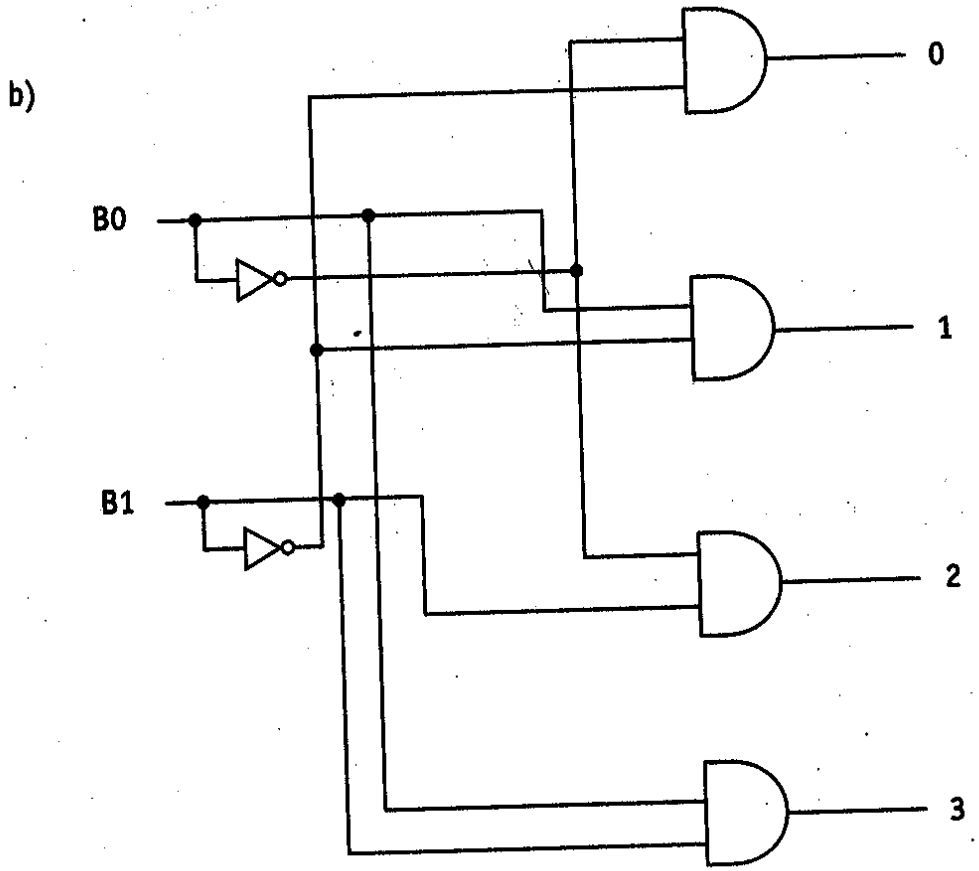
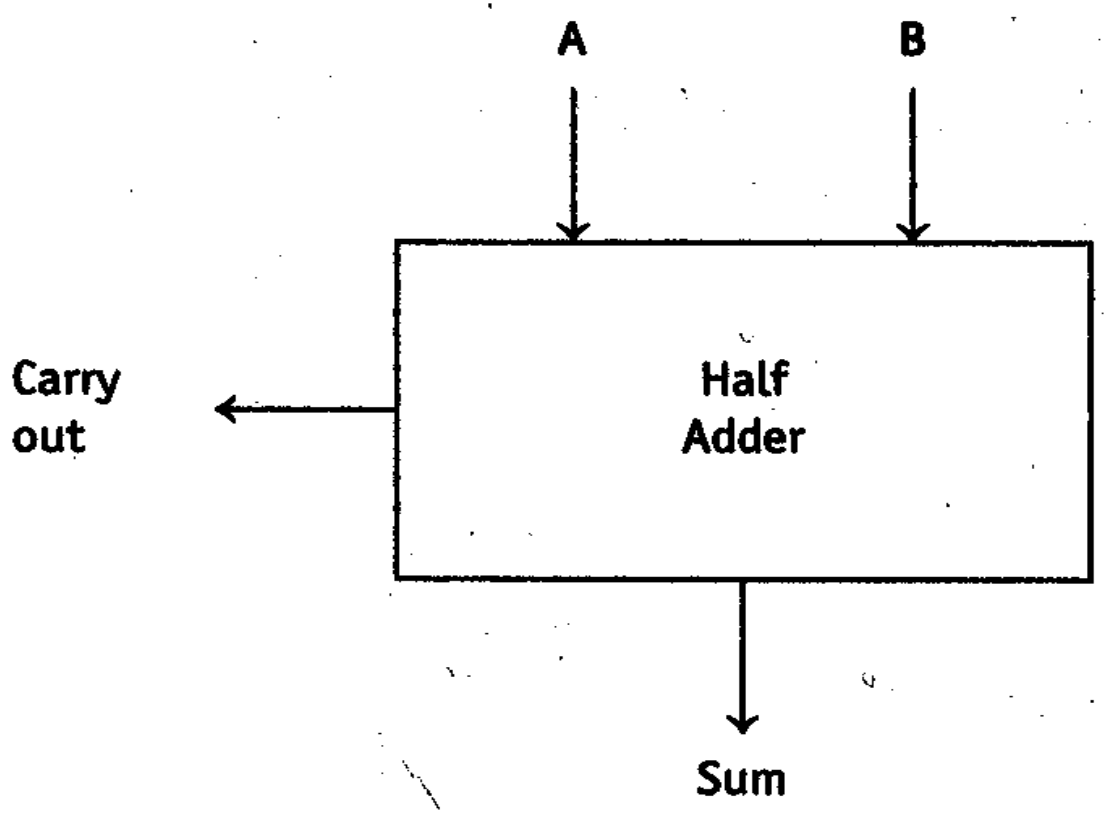


FIGURE 5.25

Half Adder

a)



b)

A	B	Sum
0	0	0
0	1	1
1	0	1
1	1	0

A	B	Carry out
0	0	0
0	1	0
1	0	0
1	1	1

Implementation of half adder

c)

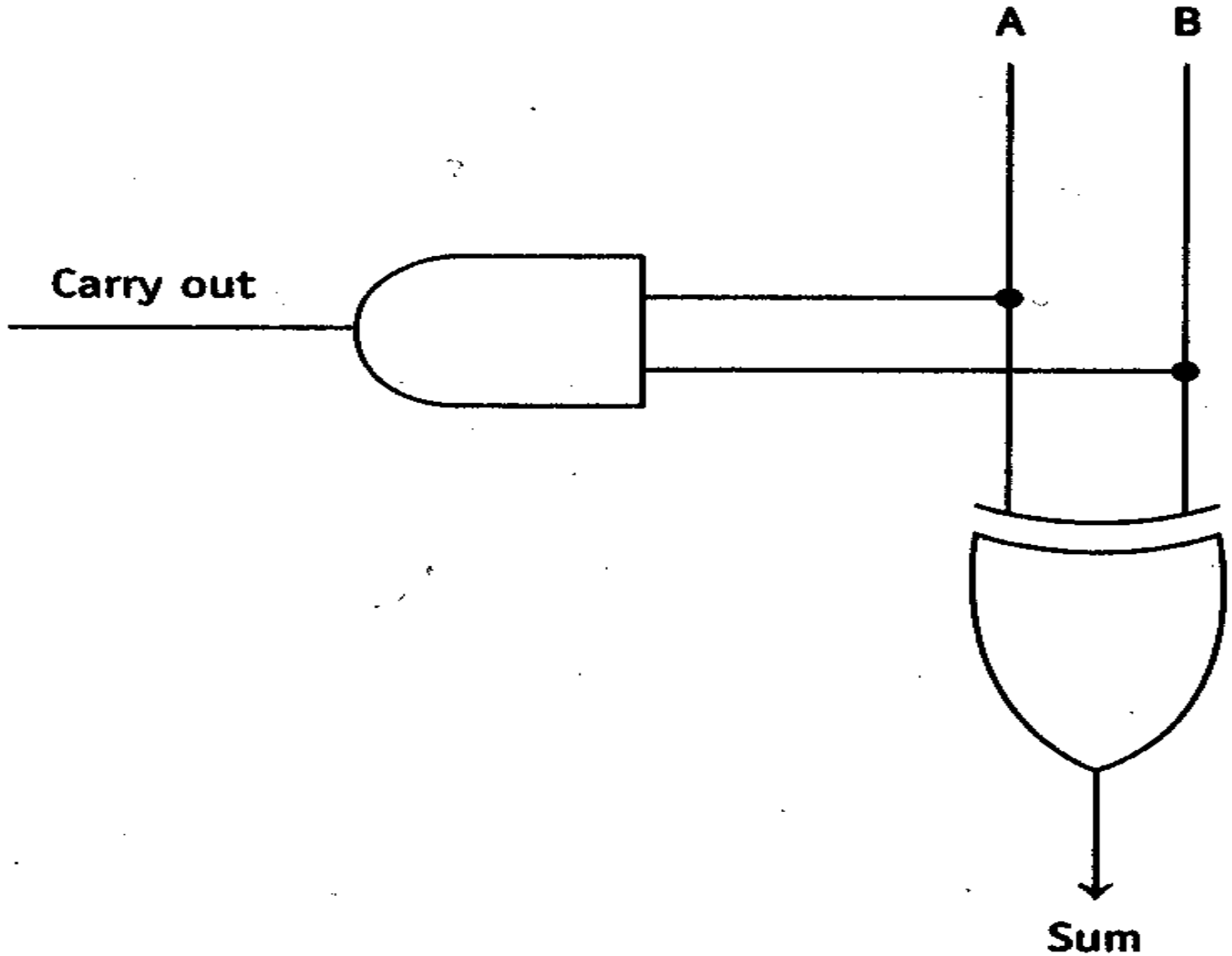


FIGURE 5.26

Full Adder

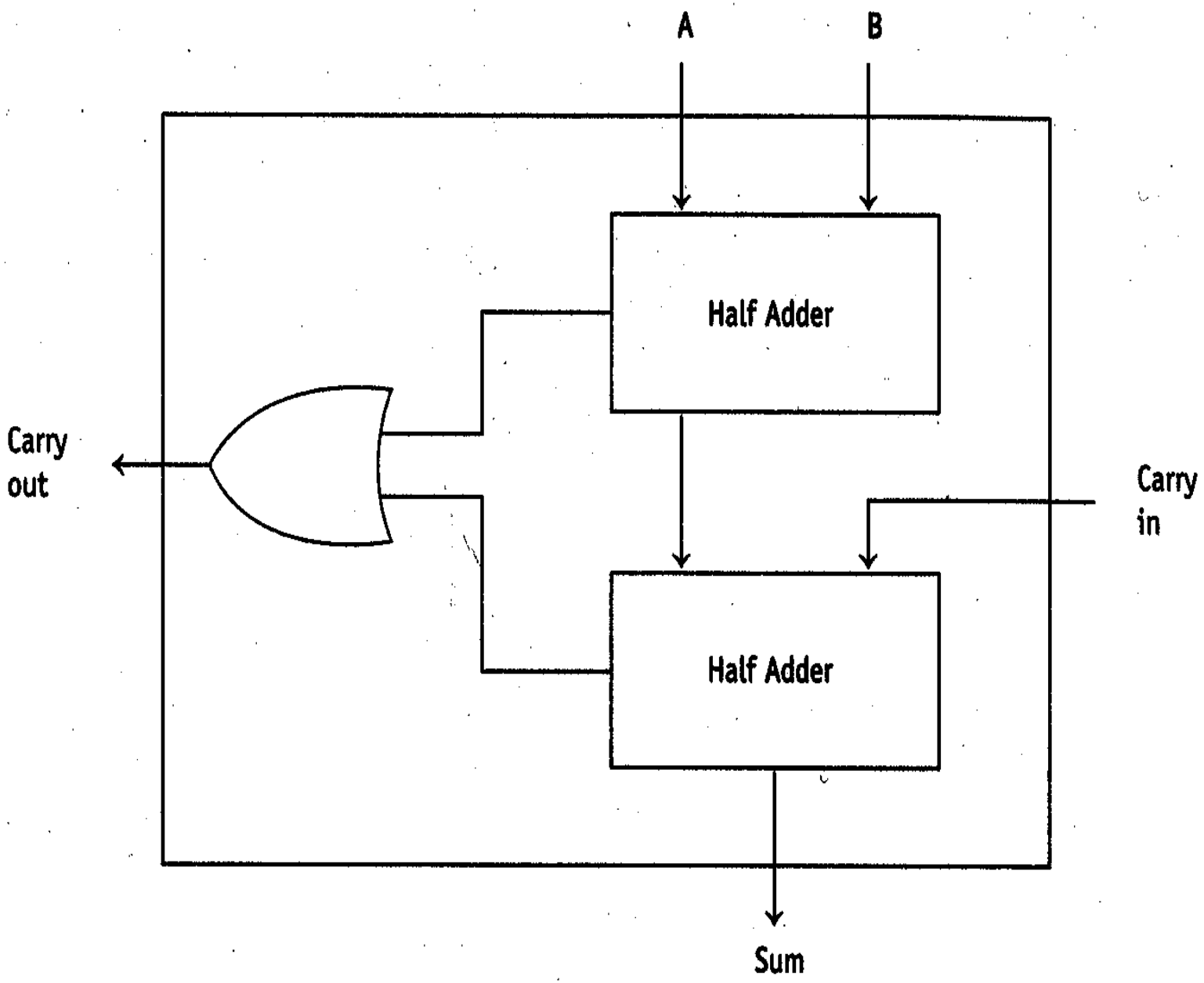
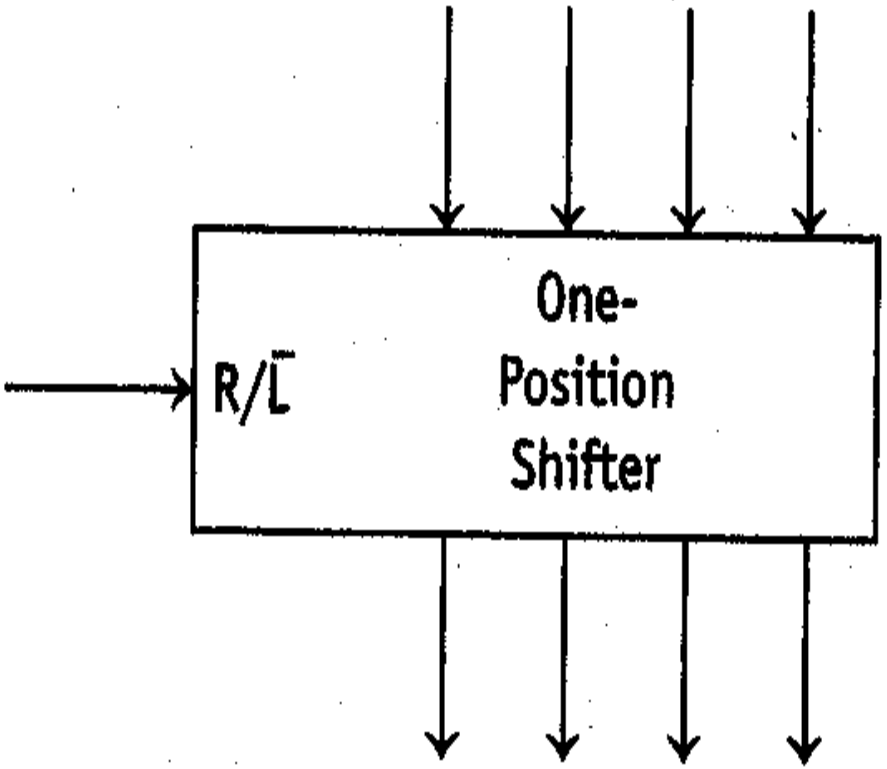


FIGURE 5.27

One-Position Shifter

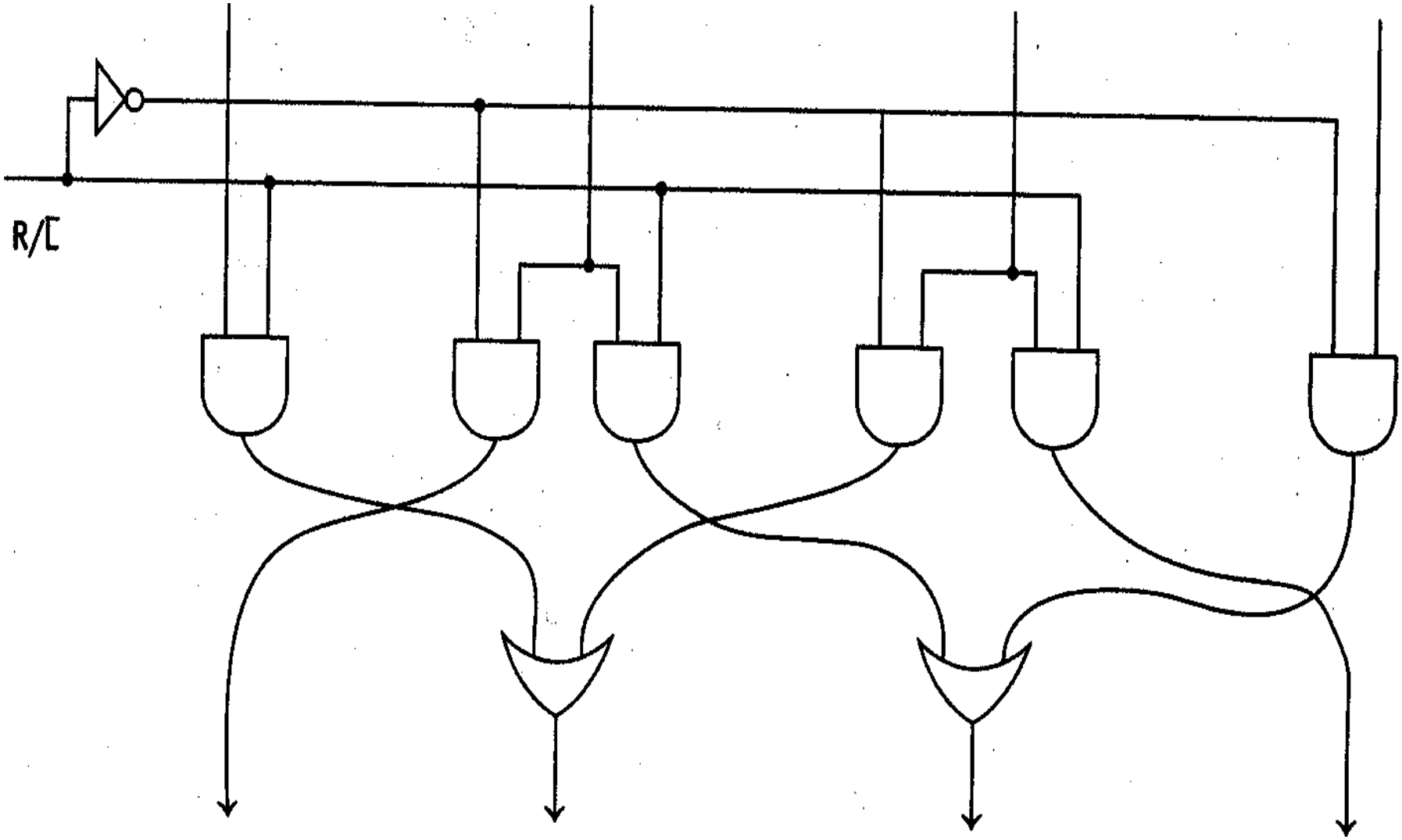
a)



b)

$R/\overline{L}$: right shift on 1, left shift on 0

Implementation of shifter



If the product of two n-bit numbers can fit into n bits, then our standard multiplication technique works for both positive and negative two’s complement numbers.

Binary multiplication

```
     1110 = −2 multiplicand
   × 1101 = −3 multiplier
   ------
     1110        partial products
    0000
   1110
  1110
  --------
 10110110        product
```

correct answer in these 4 bits

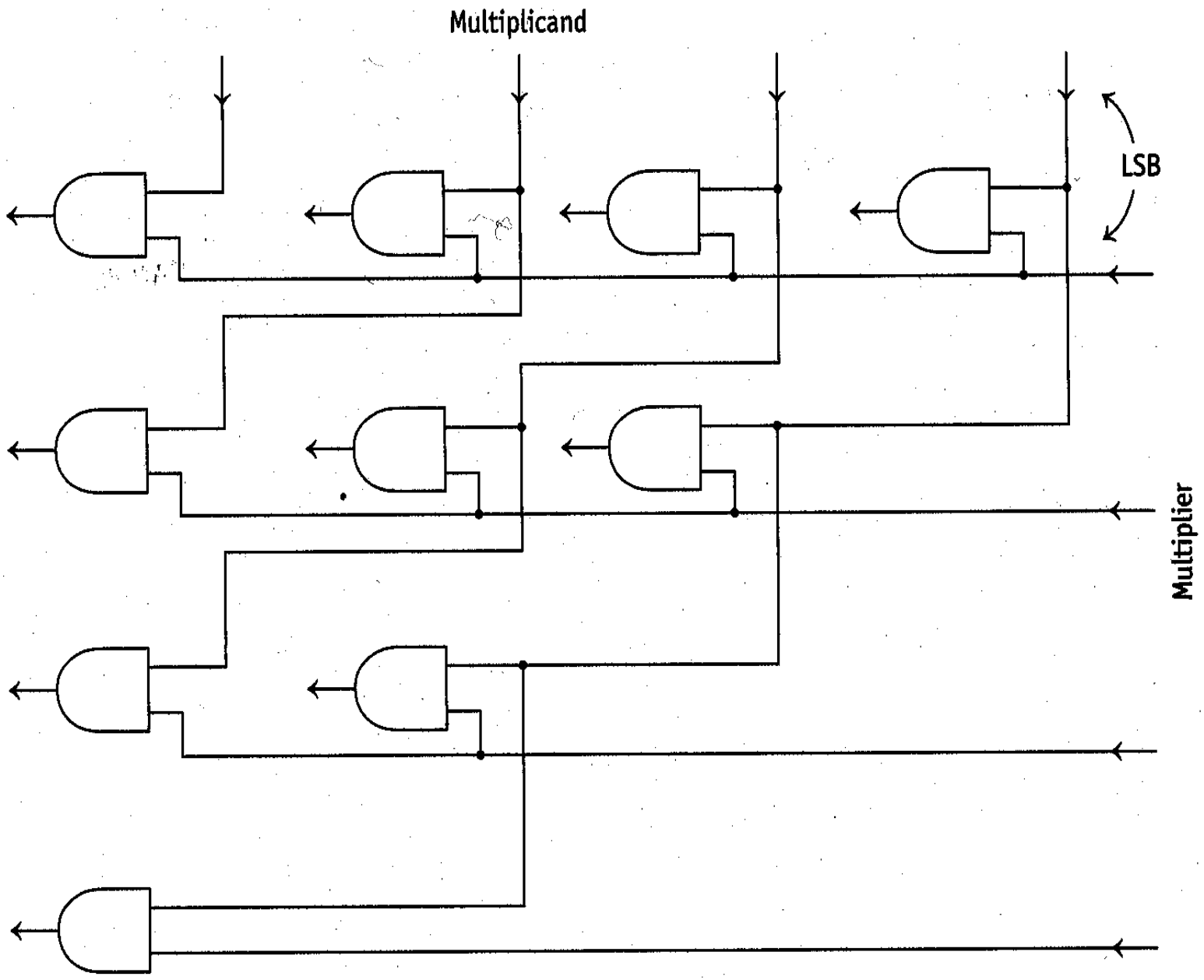
```
    1110 = −2 multiplicand
  × 0011 = +3 multiplier
  ------
    1110        partial products
   1110
  0000
 0000
 -------
 0101010        product
```

correct answer in these 4 bits

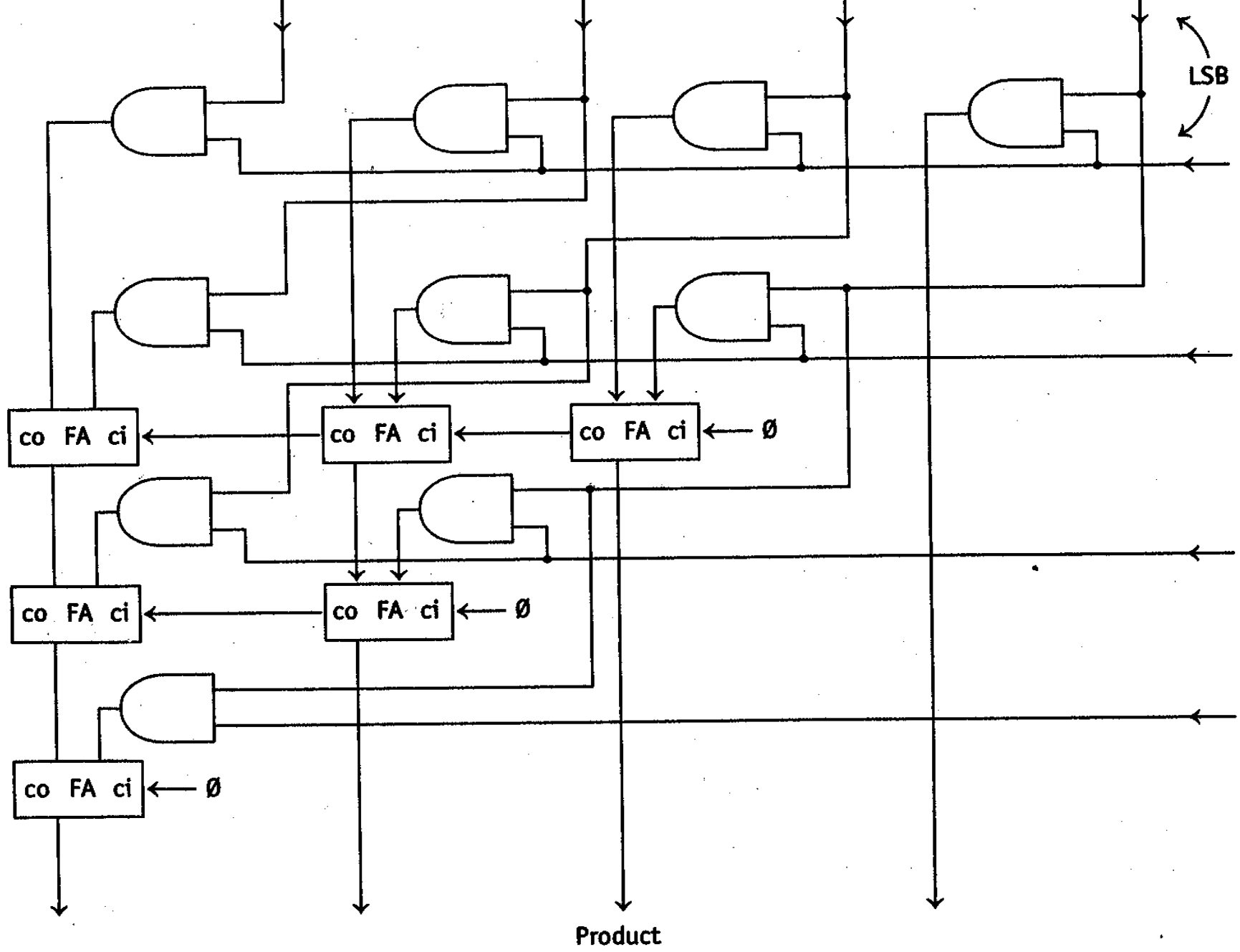
Implementing a multiplier circuit

Use AND gates to generate the partial products

FIGURE 5.28 a)

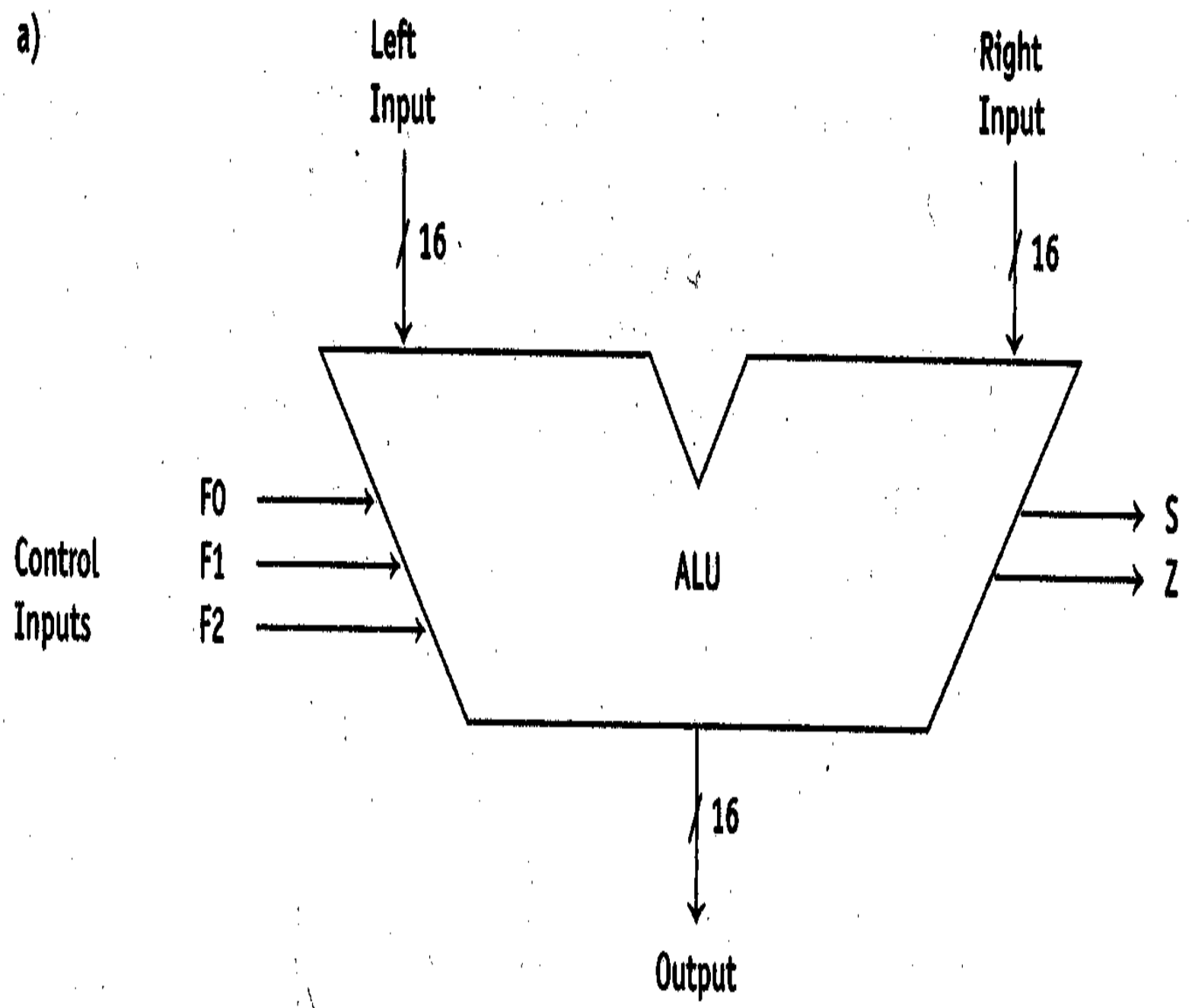


Use full adders to add corresponding bits of the partial products



Multiplier

FIGURE 5.29


a)
Left
Input
16
Right
Input
16
F0
F1
F2
Control
Inputs
ALU
S
Z
16
Output

ALU functions

b)

F2	F1	F0	Output	
0	0	0	`left`	output is same as left input
0	0	1	`~left`	bitwise complement left input
0	1	0	`left & right`	AND inputs
0	1	1	`left * right`	multiply inputs
1	0	0	`left + right`	add inputs
1	0	1	`left - right`	subtract inputs
1	1	0	`left << 1`	left shift left input one position
1	1	1	`left >> 1`	right shift left input one position

Sometimes we want data to flow through the ALU without any modification. That is the reason for function 000, which makes the ALU output identical to its left input.

Circuits needed for the ALU

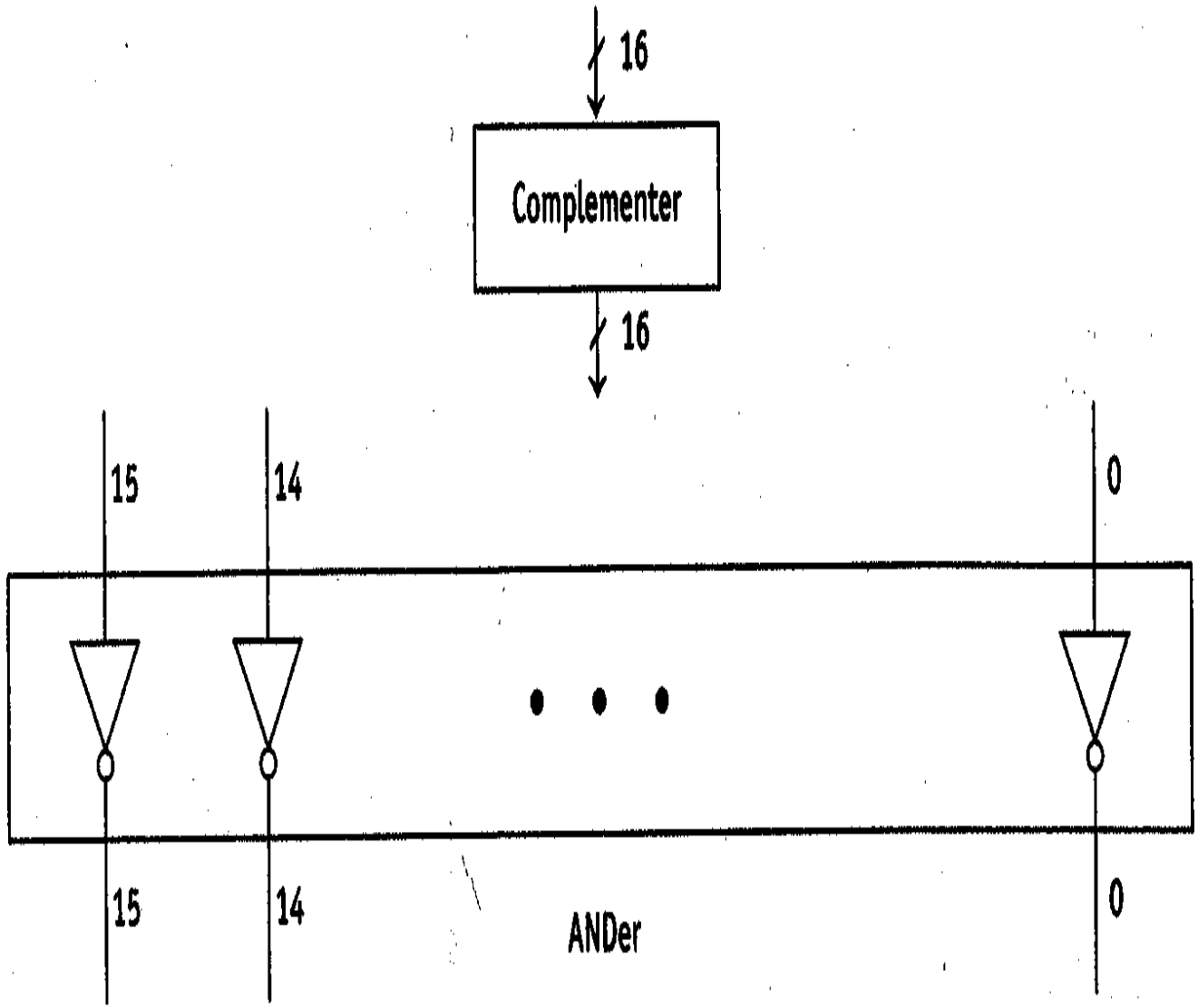
- Complementer
- ANDer
- Multiplier
- Adder/subtracter
- Shifter

We have already seen the multiplier, adder/subtracter, and shifter circuits. Let's now look at the complementer and ANDer.

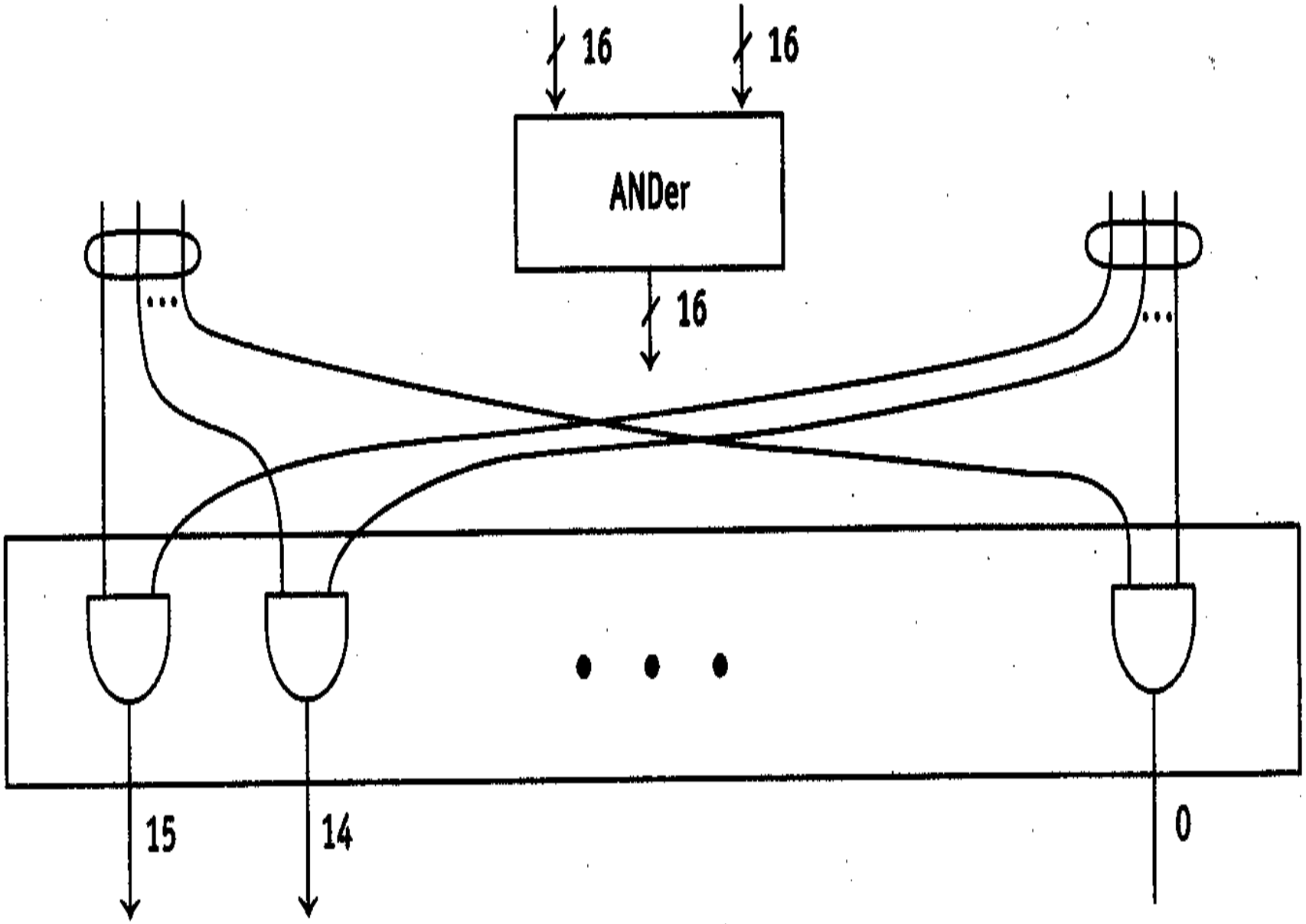
FIGURE 5.30

Complementer

a)



b)



Our ALU consists of subcircuits, each performing a single function, combined with a multiplexer.

The multiplexer selects which subcircuit output is routed to the ALU output.

FIGURE 5.31

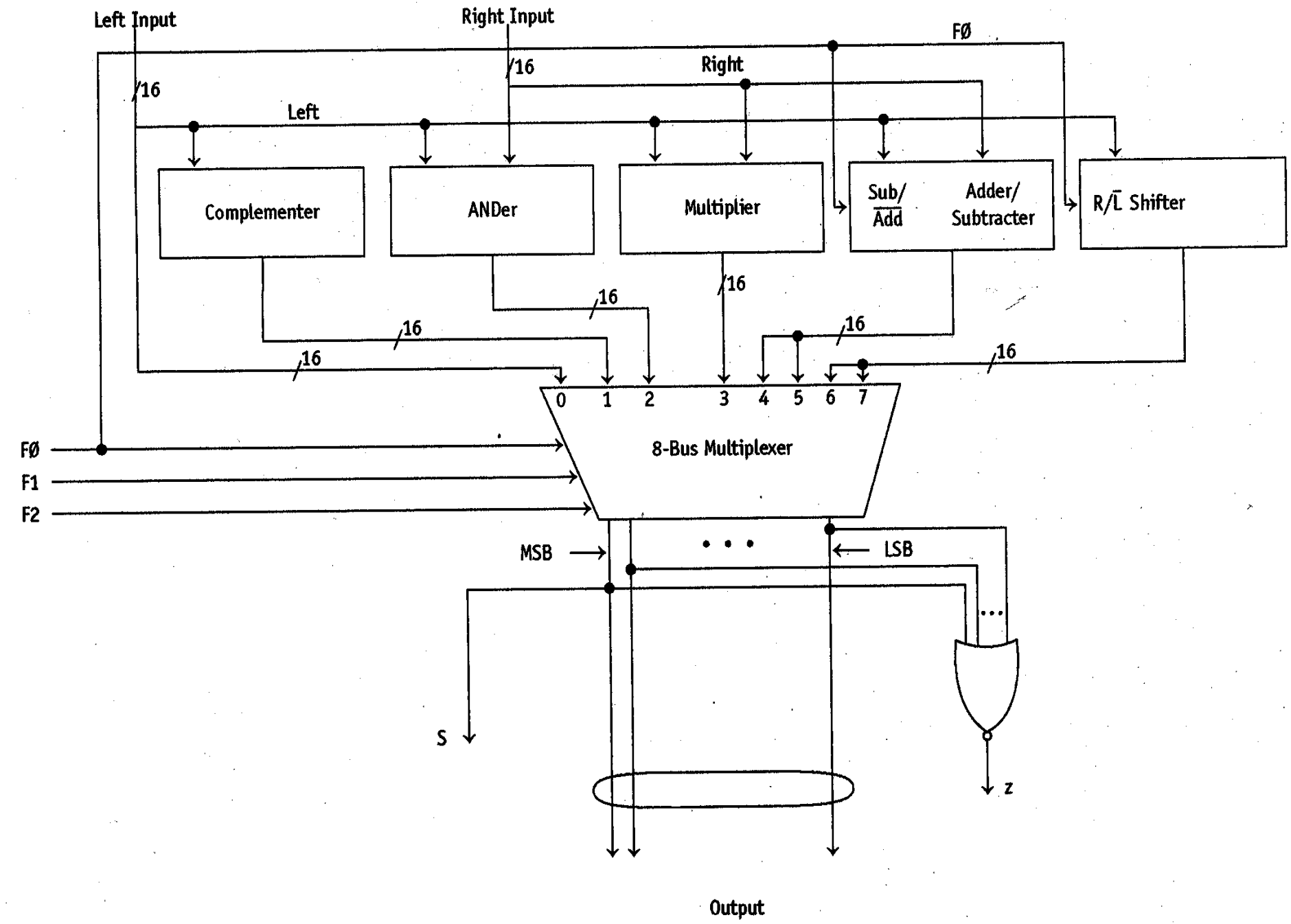
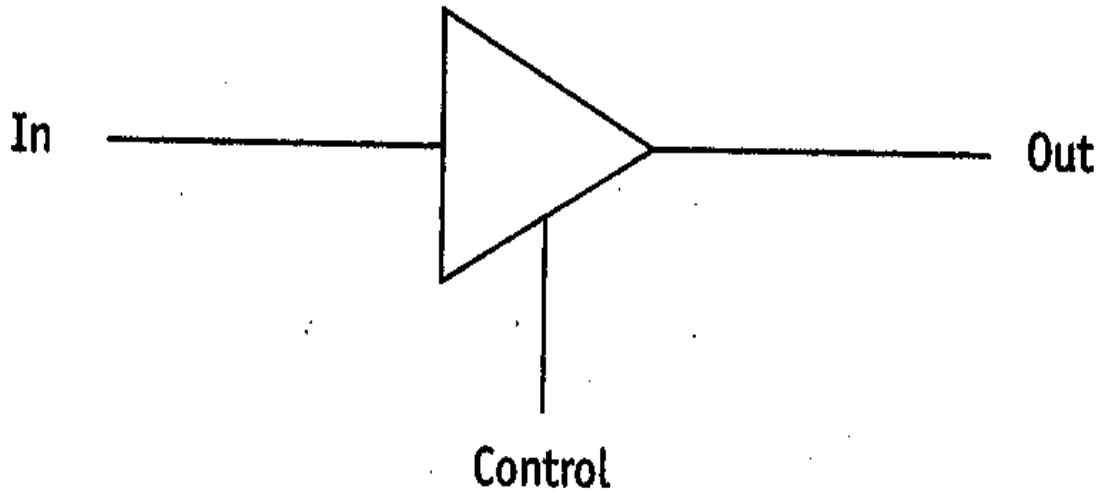

Left Input
Right Input
FØ
16
Right
Left
Complementer
ANDer
Multiplier
Sub/
Add
Adder/
Subtracter
R/L̄ Shifter
16
16
16
16
16
16
0
1
2
3
4
5
6
7
8-Bus Multiplexer
FØ
F1
F2
MSB
LSB
S
z
Output

FIGURE 5.33

Tri-State Buffer

a)

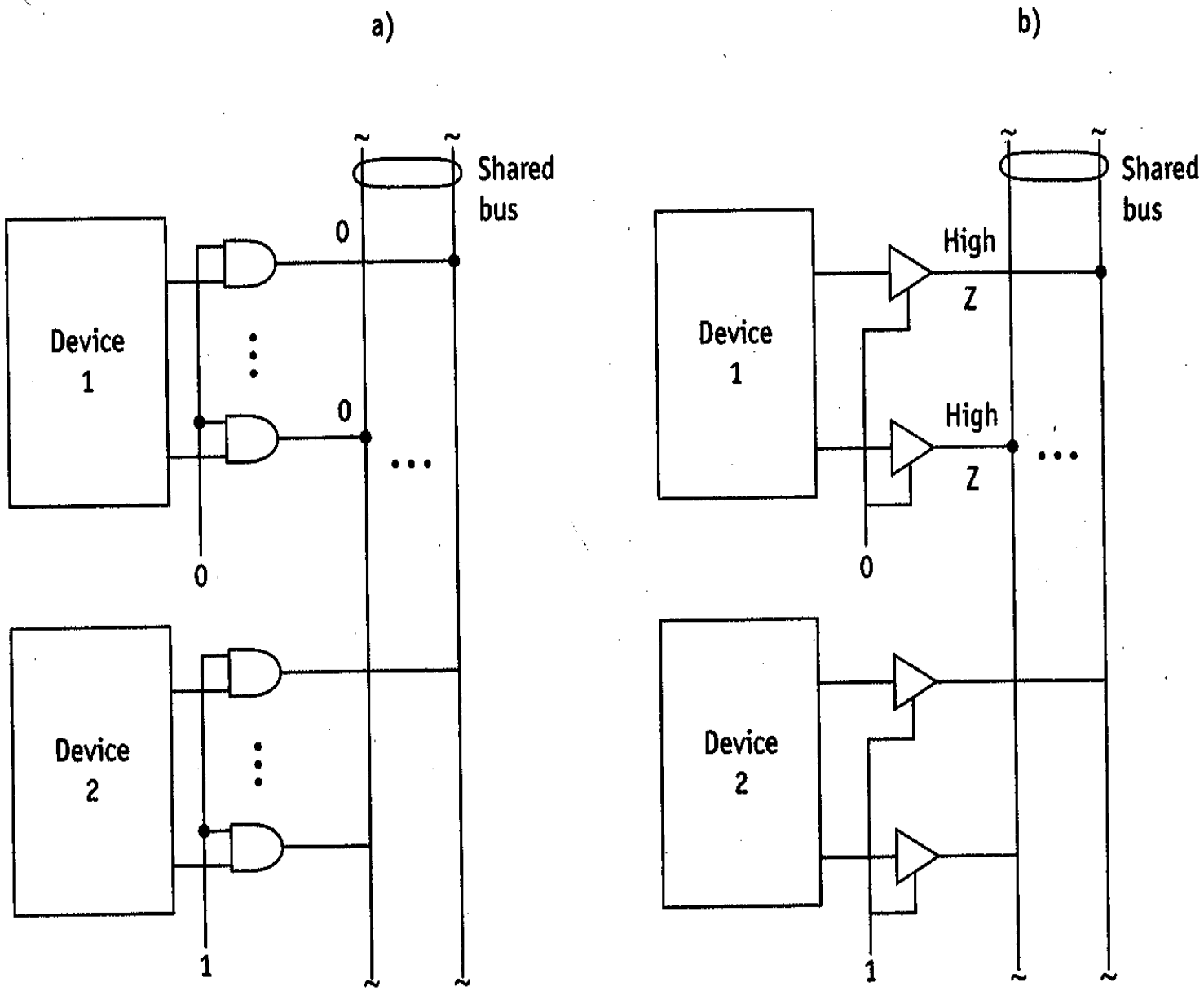


b)

Control	In	Out
0	0	High Z
0	1	High Z
1	0	0
1	1	1

FIGURE 5.32

Shared Bus



Implementation of tri-state buffer

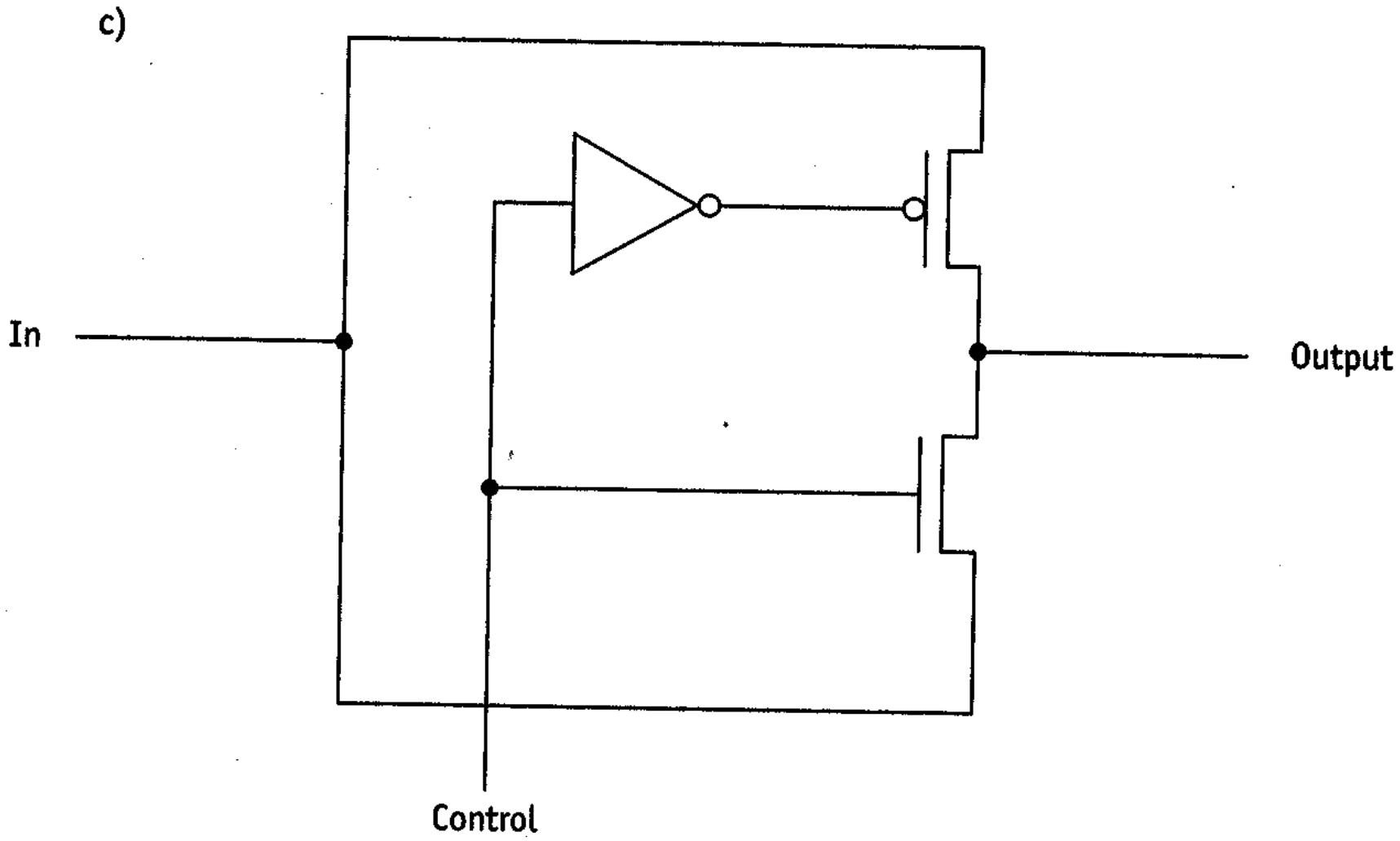
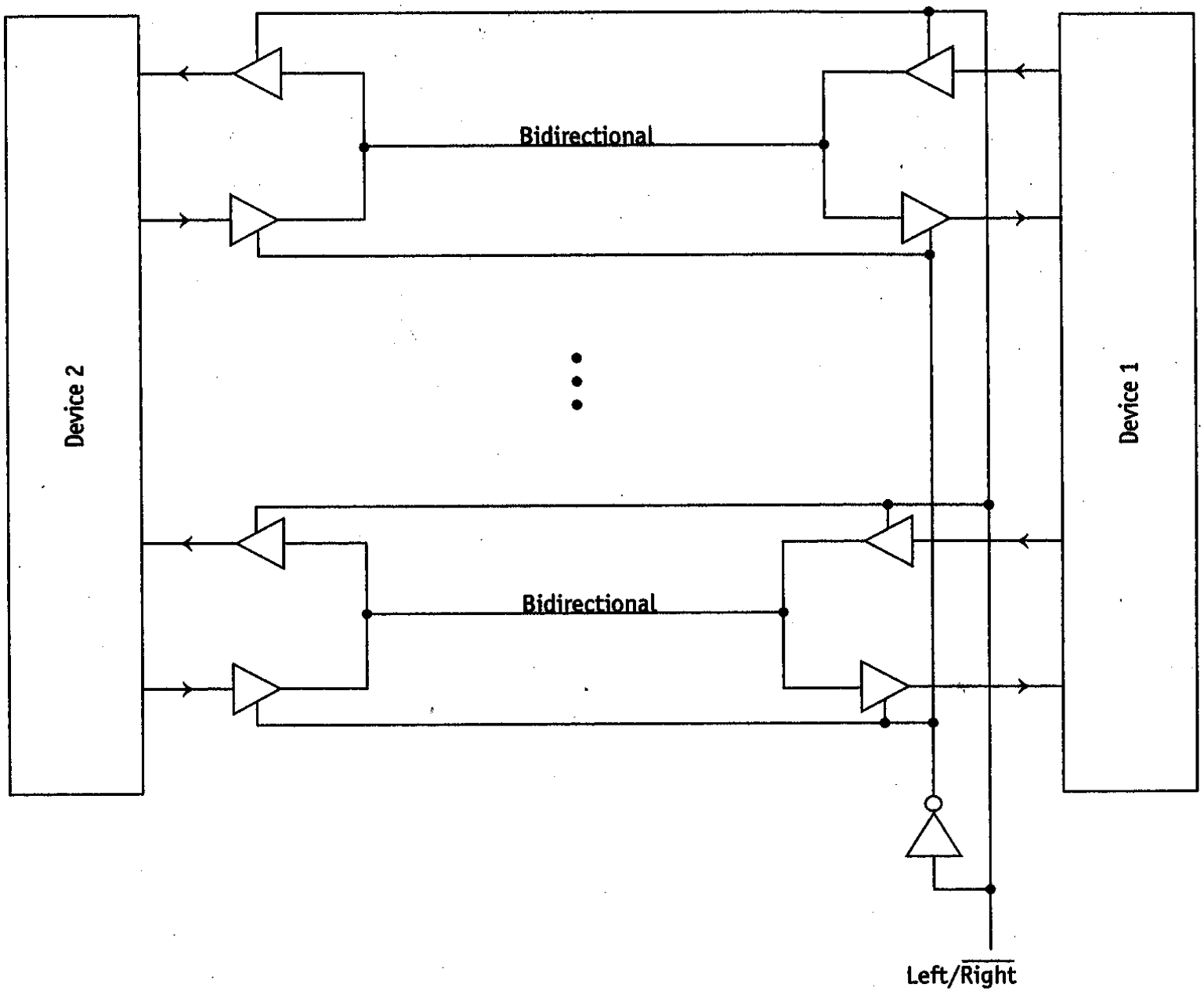
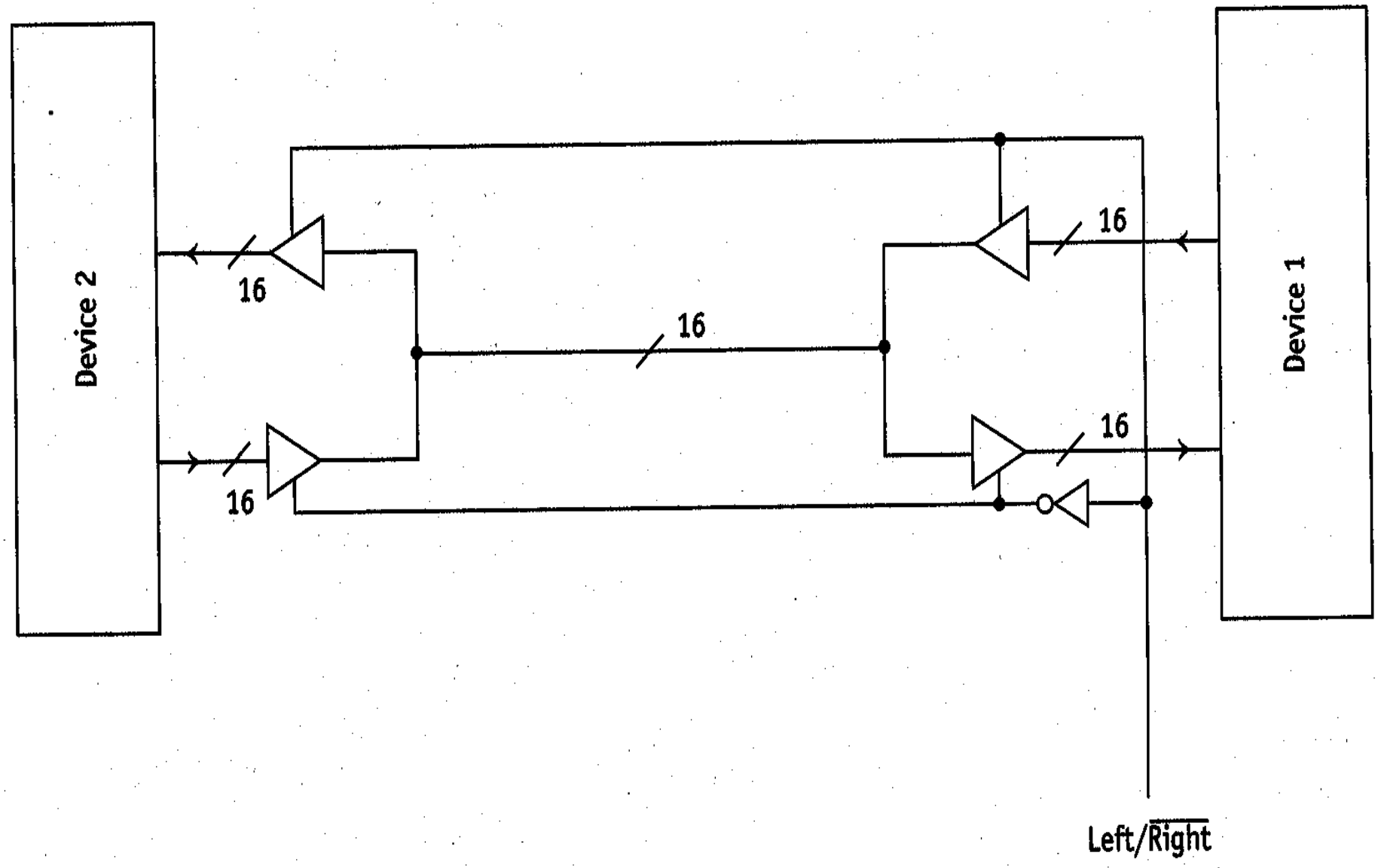


FIGURE 5.34 a)



b)

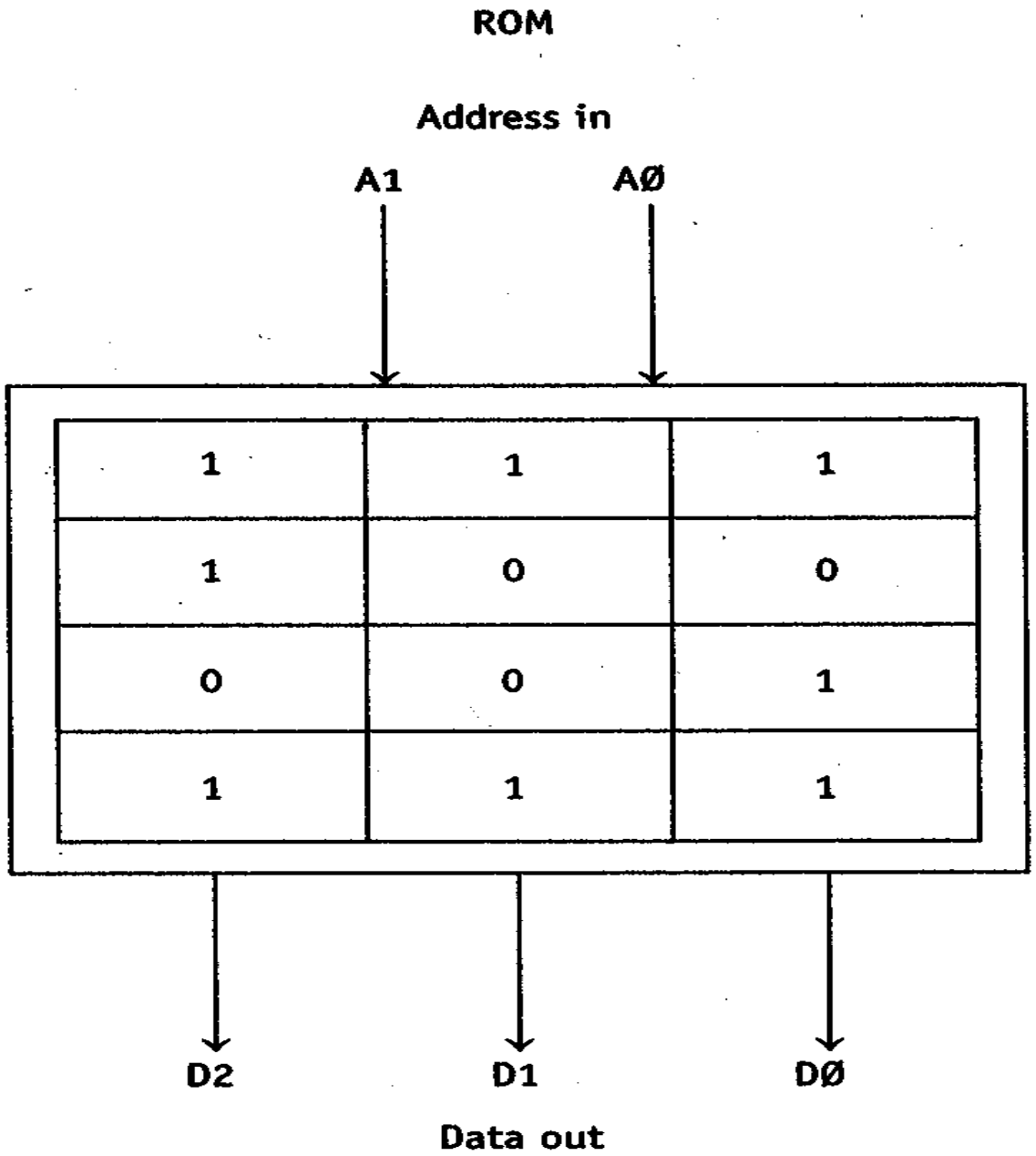
Simplified diagram of bidirectional bus



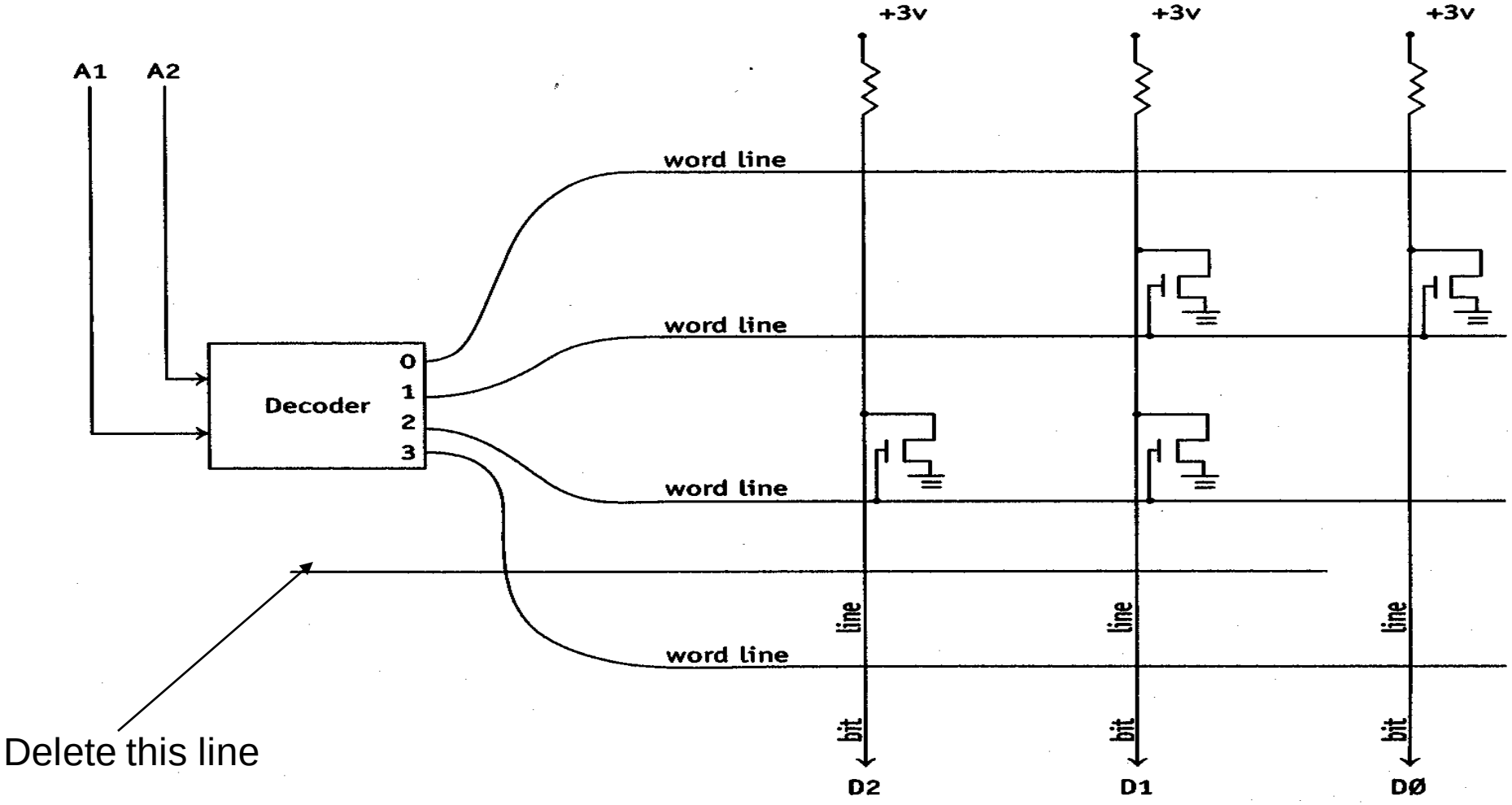
ROM – read-only memory
RAM – read/write memory

Both ROM and RAM are “random access” memories.

FIGURE 5.35 a)



Implementation of ROM



Sequential Circuits

Circuits in which the present output depends on past and present inputs. Sequential circuits have memory.

Flip-flop

Two-state device that can hold 1 bit.

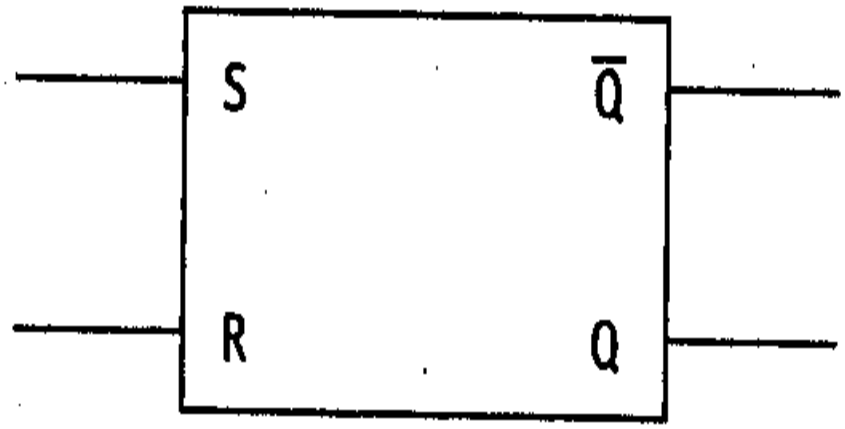
When *set*, a flip-flop's Q output is 1. When *reset*, its Q output is 0.

SR flip-flop and S and R inputs

FIGURE 5.36

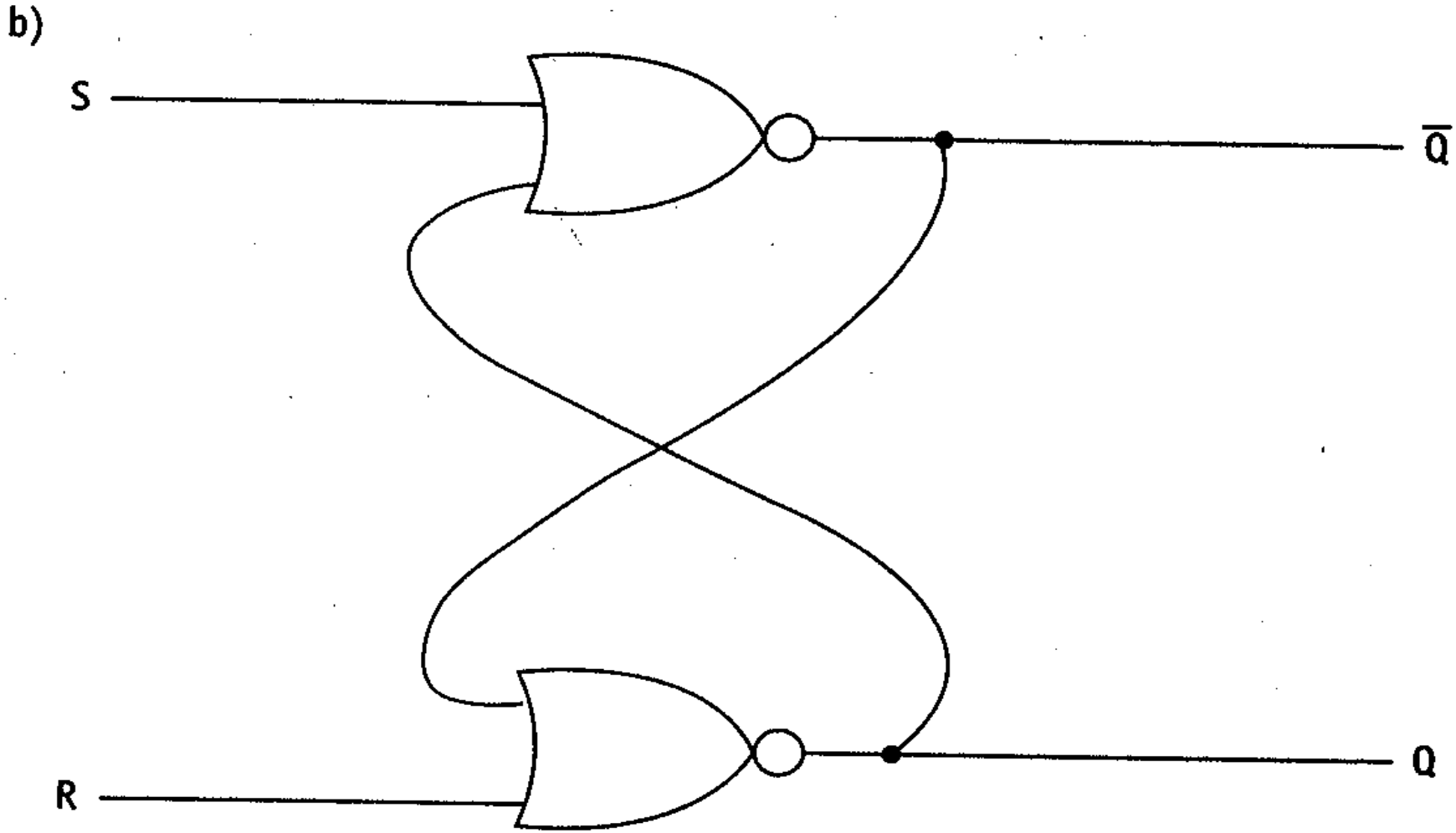
SR Flip-Flop
(NOR Version)

a)



Flip-flops use *feedback* (an output line is fed back into an input).

Implementation of SR flip-flop



Truth table for SR flip-flop.

Two states are possible when S and R are both 0. Which state depends on the previous input.

c)

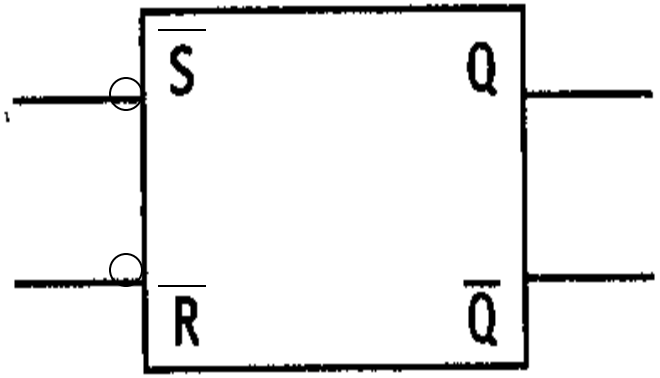
S	R	Q	$\overline{Q}$
0	1	0	1
1	0	1	0
1	1	0	0
0	0	1	0
0	0	0	1

SR flip-flop can be implemented with NORs or NANDS

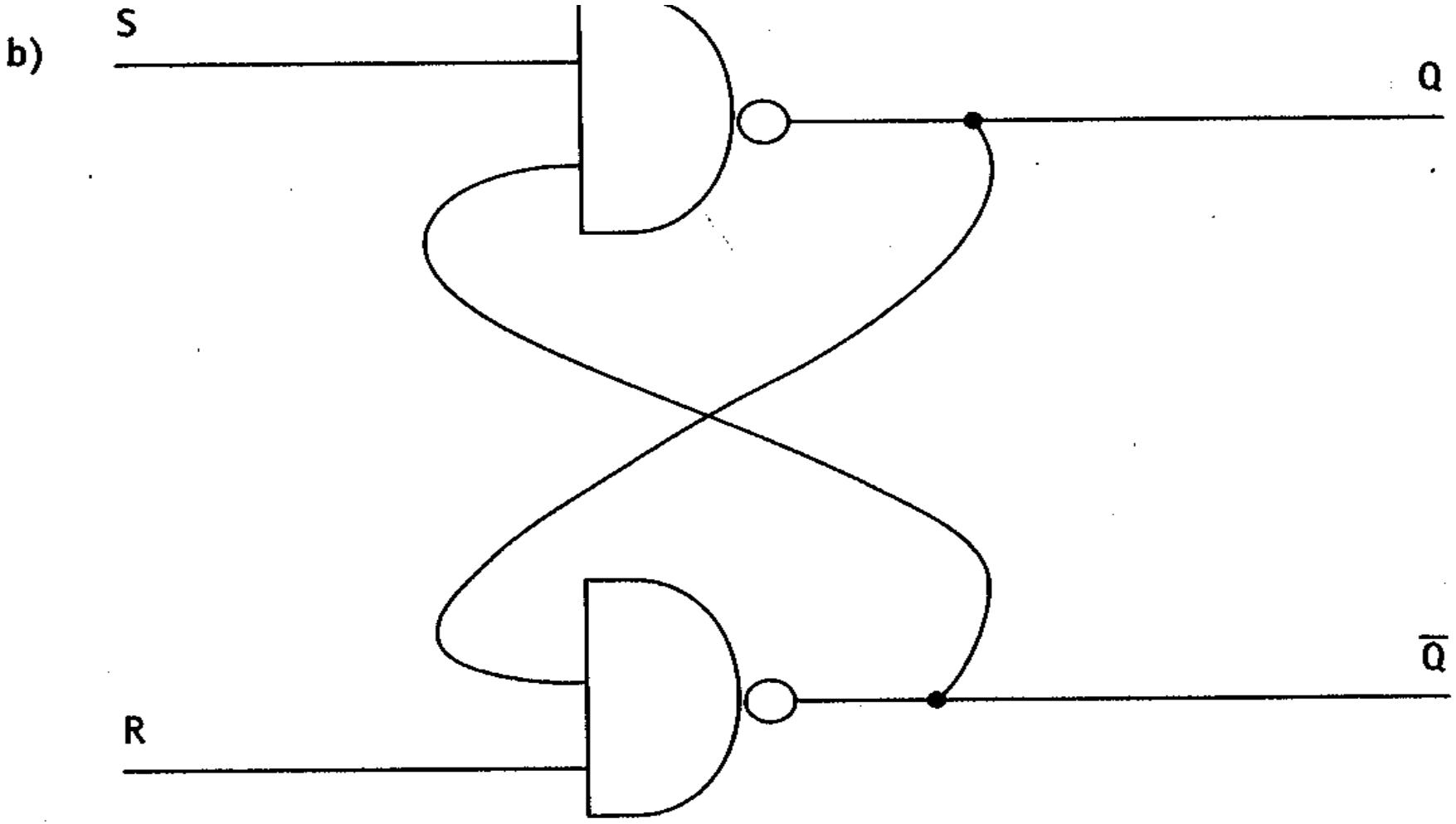
FIGURE 5.37

SR Flip-Flop
(NAND Version)

a)



Implementation of SR flip-flop—NAND version



Truth table for SR flip-flop NAND version

Role of 0 and 1 is reversed.

Two states are possible when S and R are both 1.

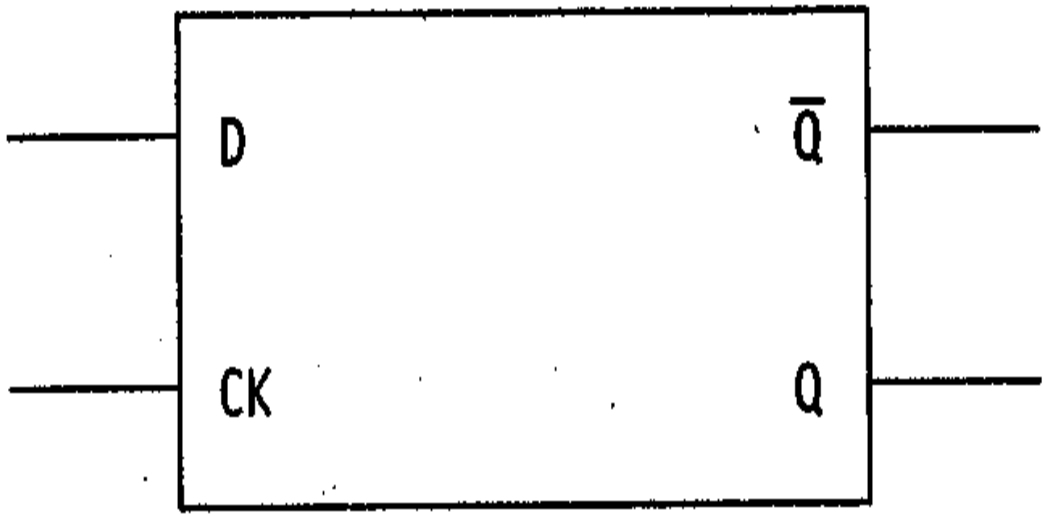
c)

S	R	Q	$\overline{Q}$
0	0	1	1
0	1	1	0
1	0	0	1
1	1	0	1
1	1	1	0

The clock input determines when a flip-flop will respond to its inputs. It is often driven by an clock.

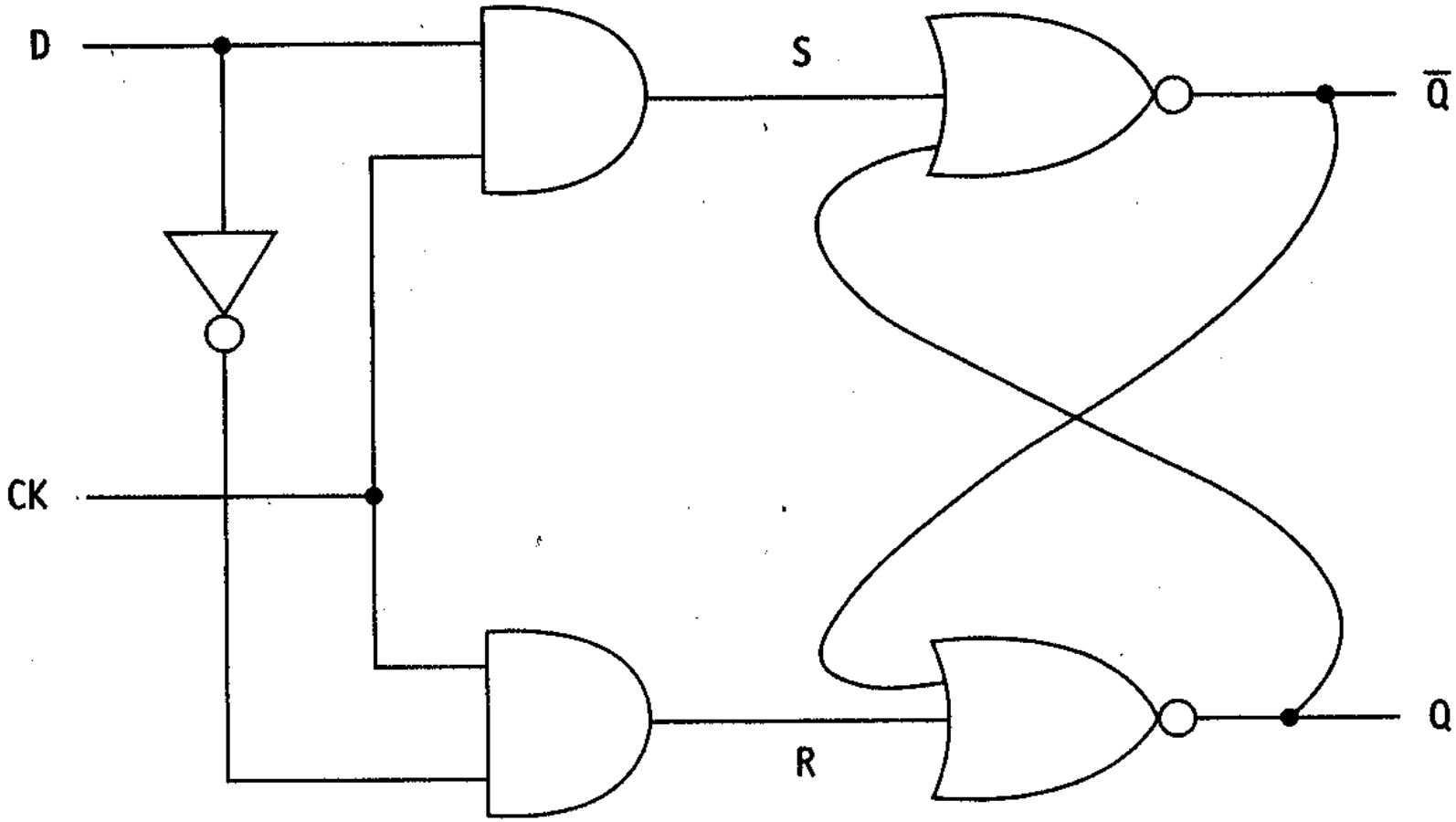
FIGURE 5.38 Clocked D Flip-Flop

a)



Implementation of clocked D flip-flop

b)

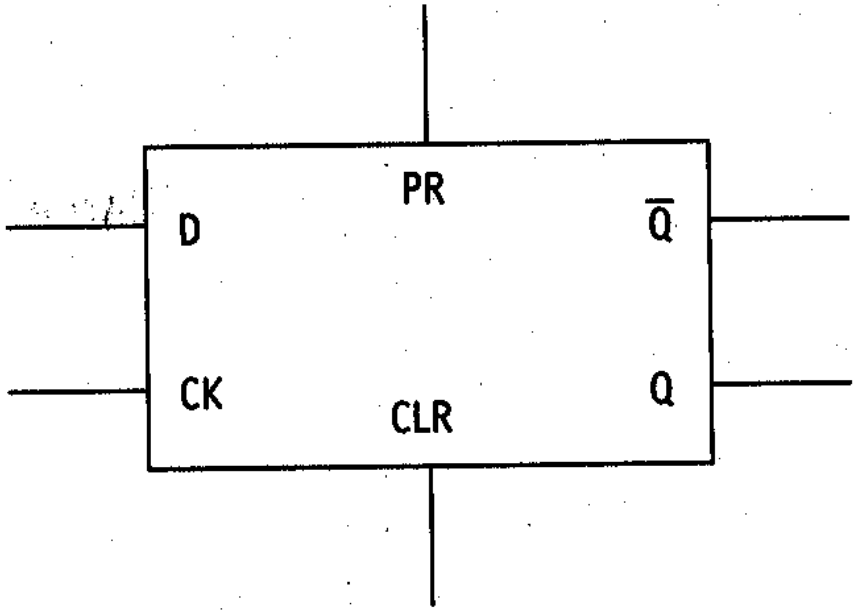


PR and CLR allows setting/resetting at any time

FIGURE 5.39

Clocked D Flip-Flop with PR & CLR
(NOR Version)

a)



Implementation of clocked D flip-flop with PR and CLR

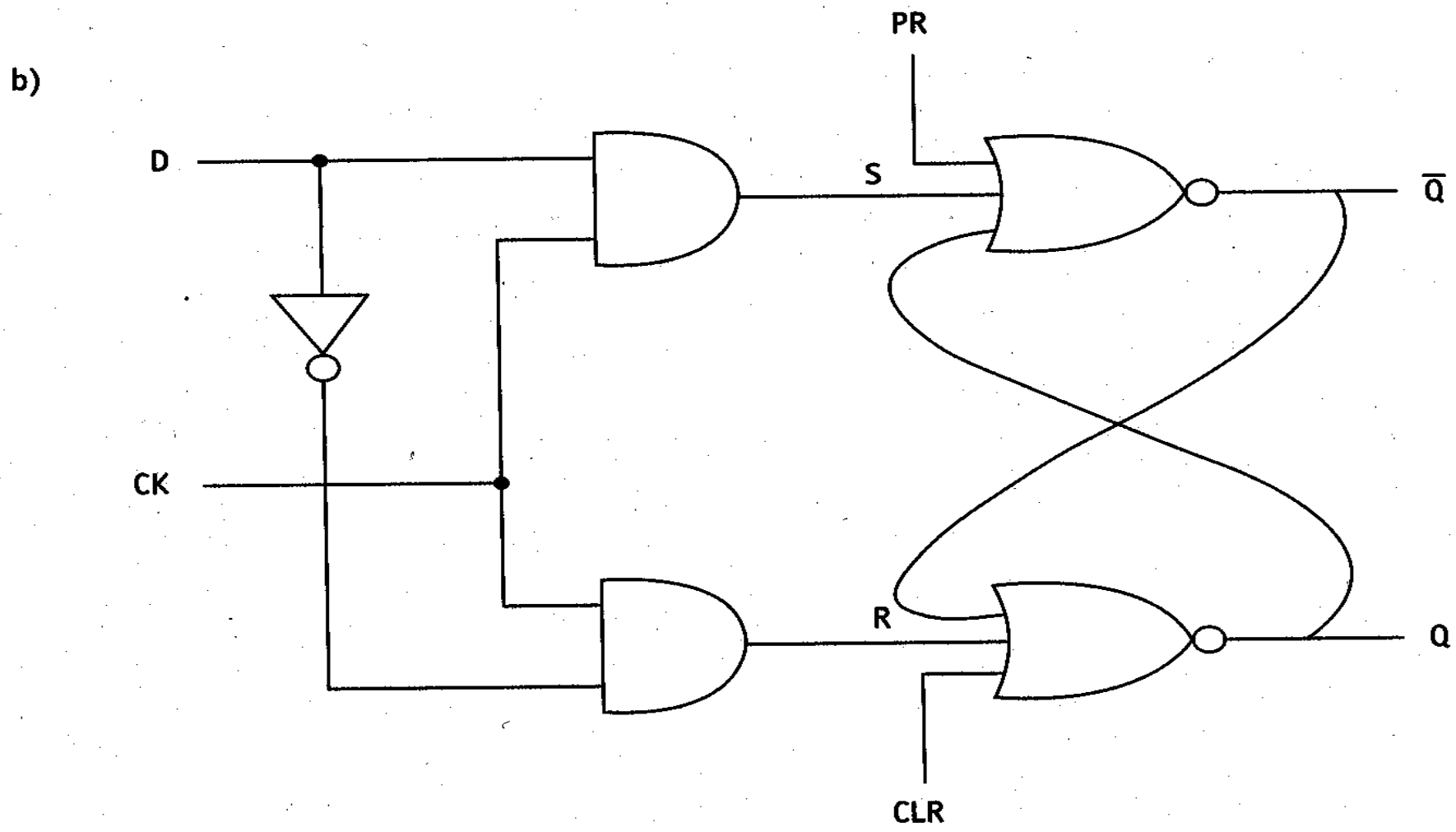
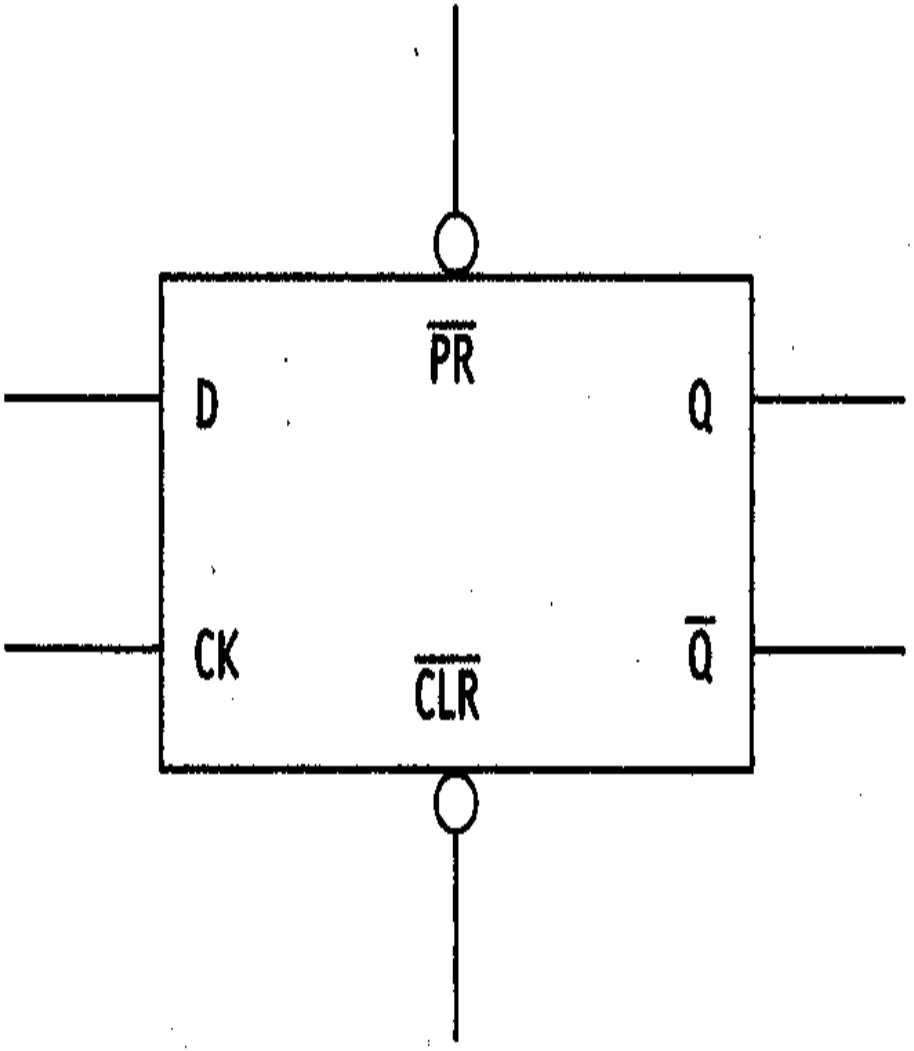


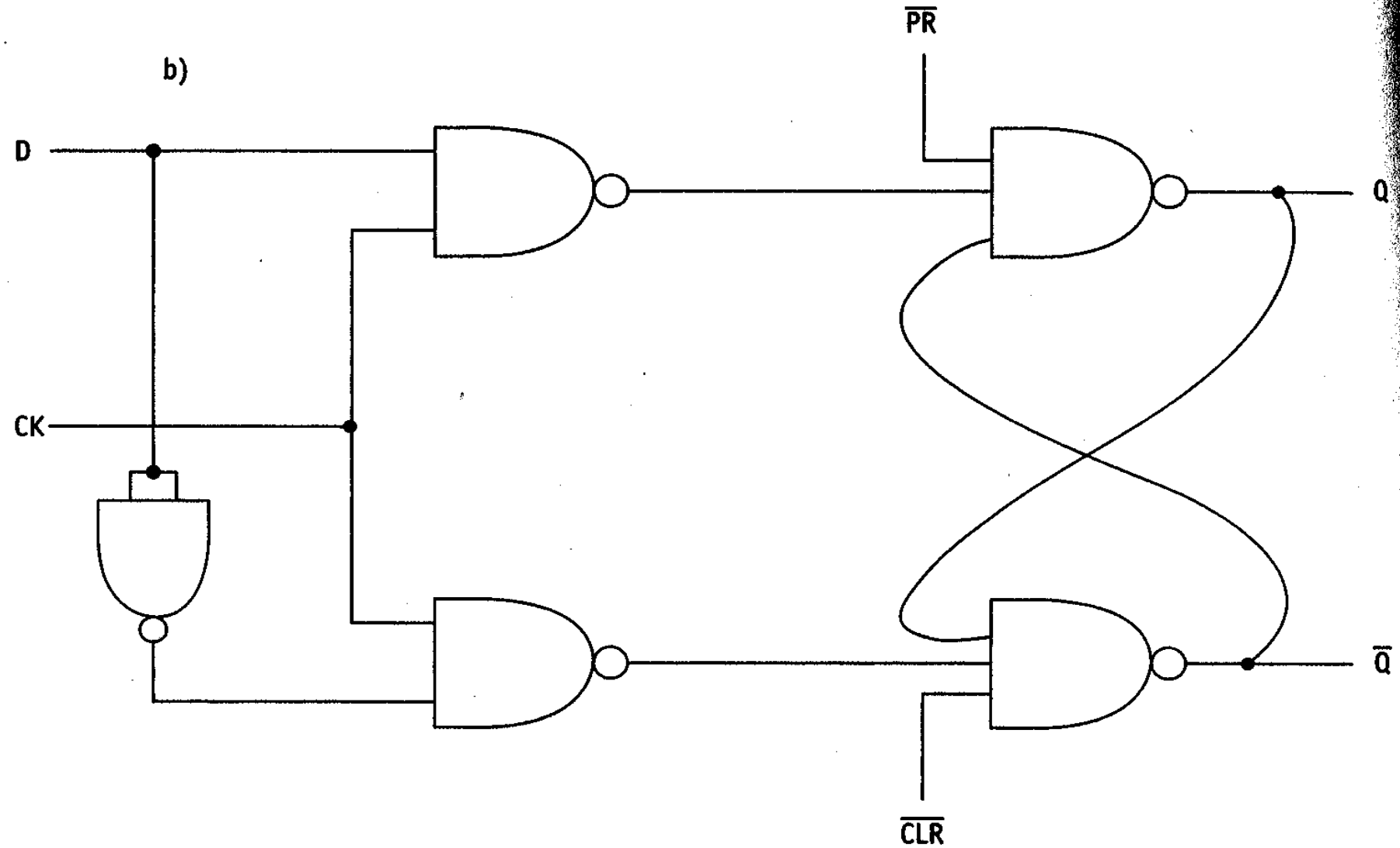
FIGURE 5.40

Clocked D Flip-Flop with PR & CLR
(NAND Version)

a)



Implementation of clocked D flip-flop with PR and CLR

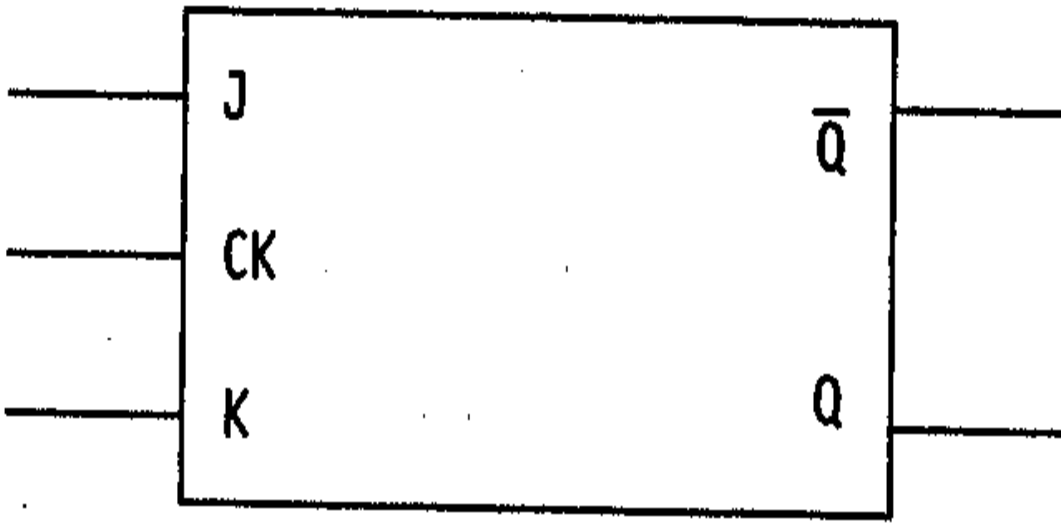


JK flip-flop changes state when J= K = 1 and CK is asserted

FIGURE 5.41

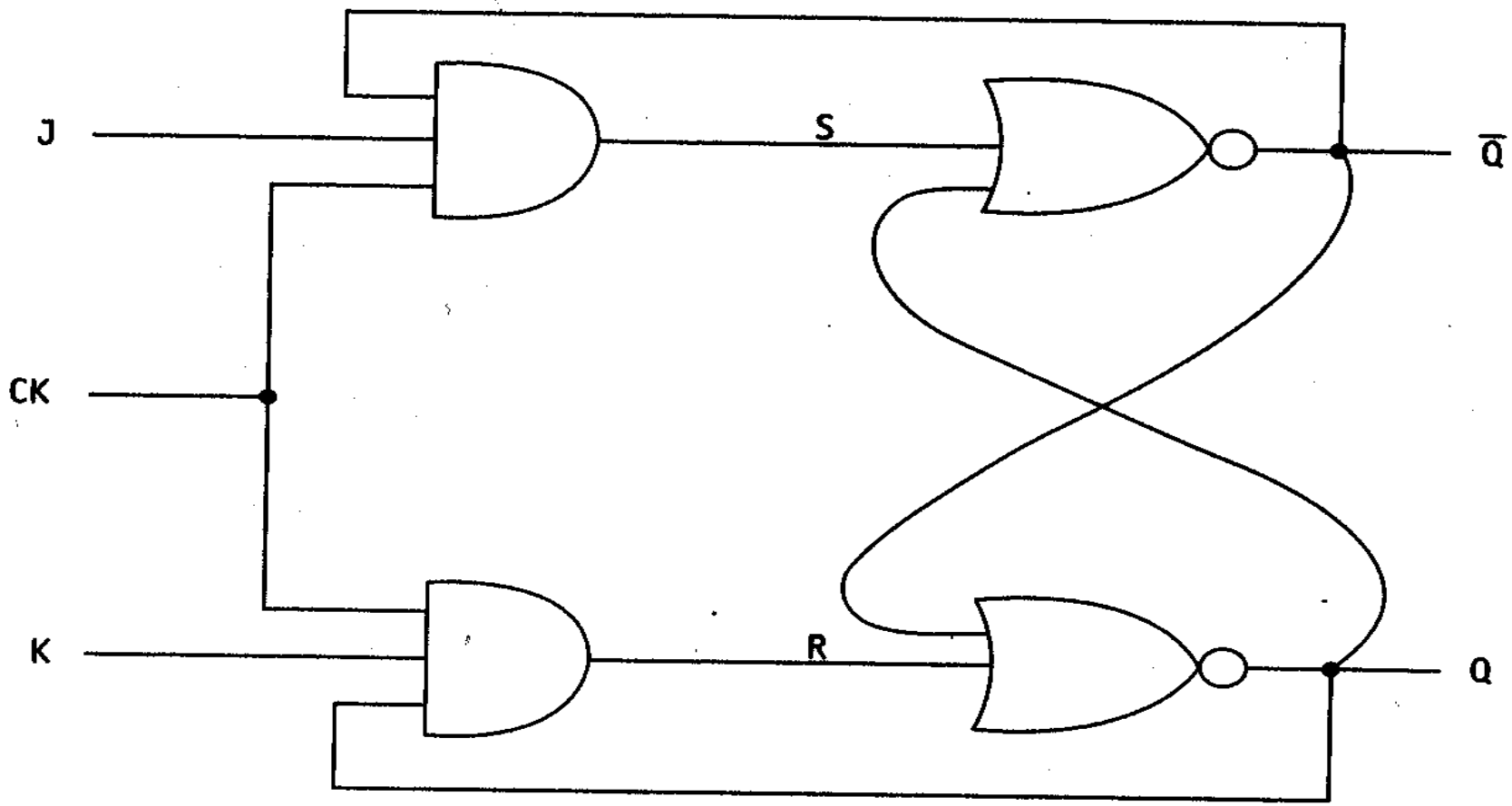
Clocked JK Flip-Flop

a)



Implementation of JK flip-flop

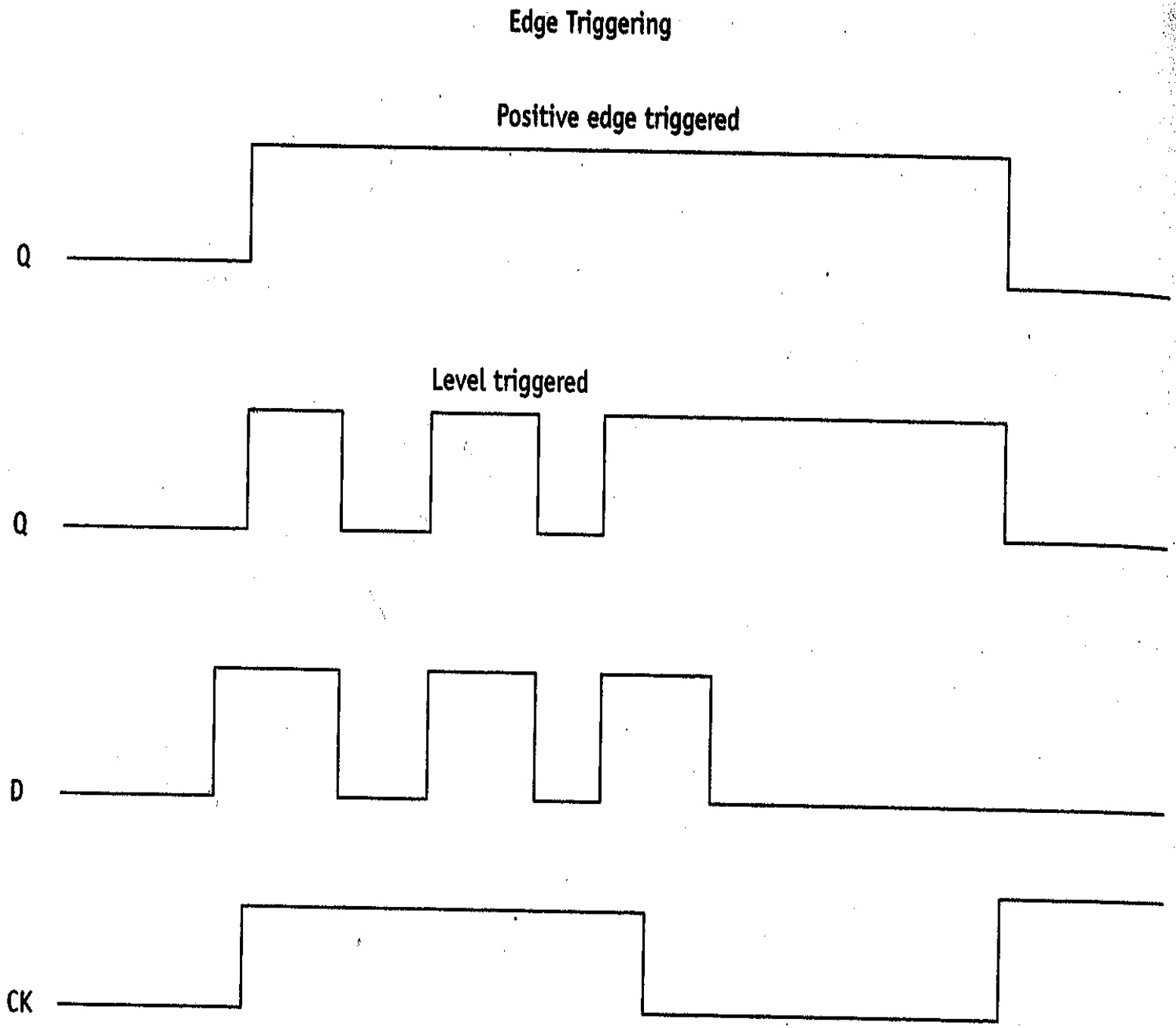
b)



Level-triggered flip-flop
Responds to input whenever the CK input is asserted.

Edge-triggered flip-flop
Responds to the input only on an *edge* (i.e., transition) of the CK input.

FIGURE 5.42

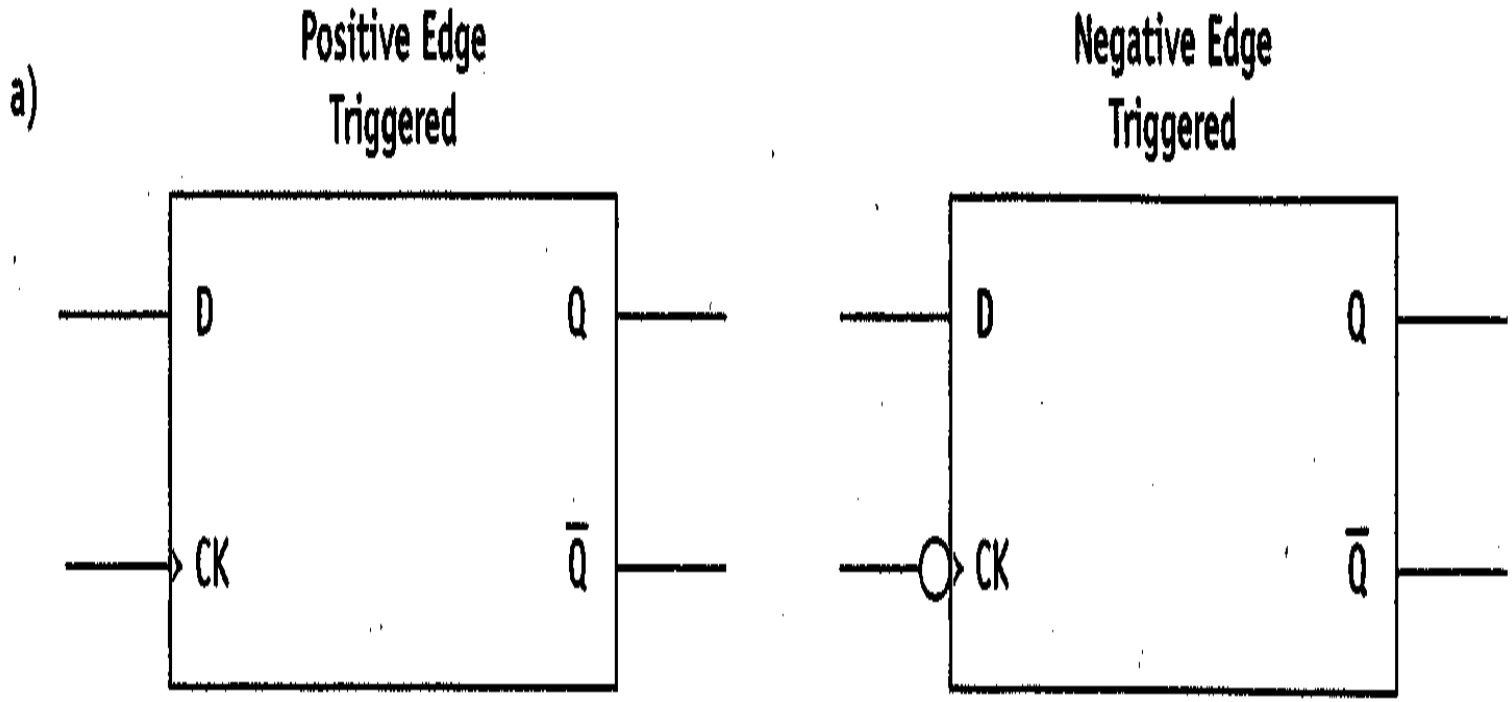

Edge Triggering
Positive edge triggered
Q
Level triggered
Q
D
CK

Triggering digital circuits

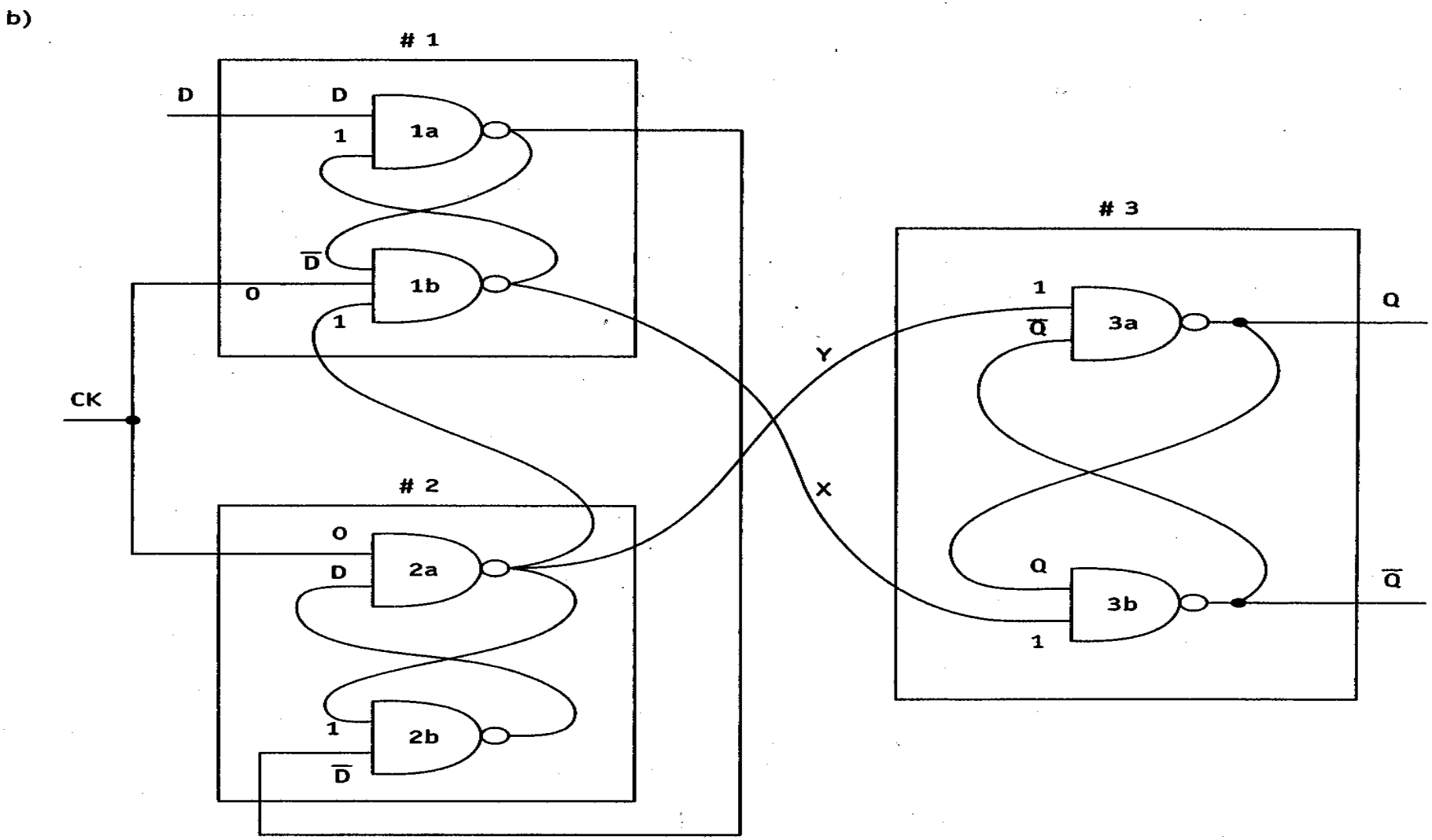
- Level triggered: circuit responds whenever CK is asserted.
- Positive-edge triggered: circuit responds only during a positive-going input to CK.
- Negative-edge triggered: circuit responds only during a negative-going input to CK.

‘>’ indicates edge-triggering

FIGURE 5.43

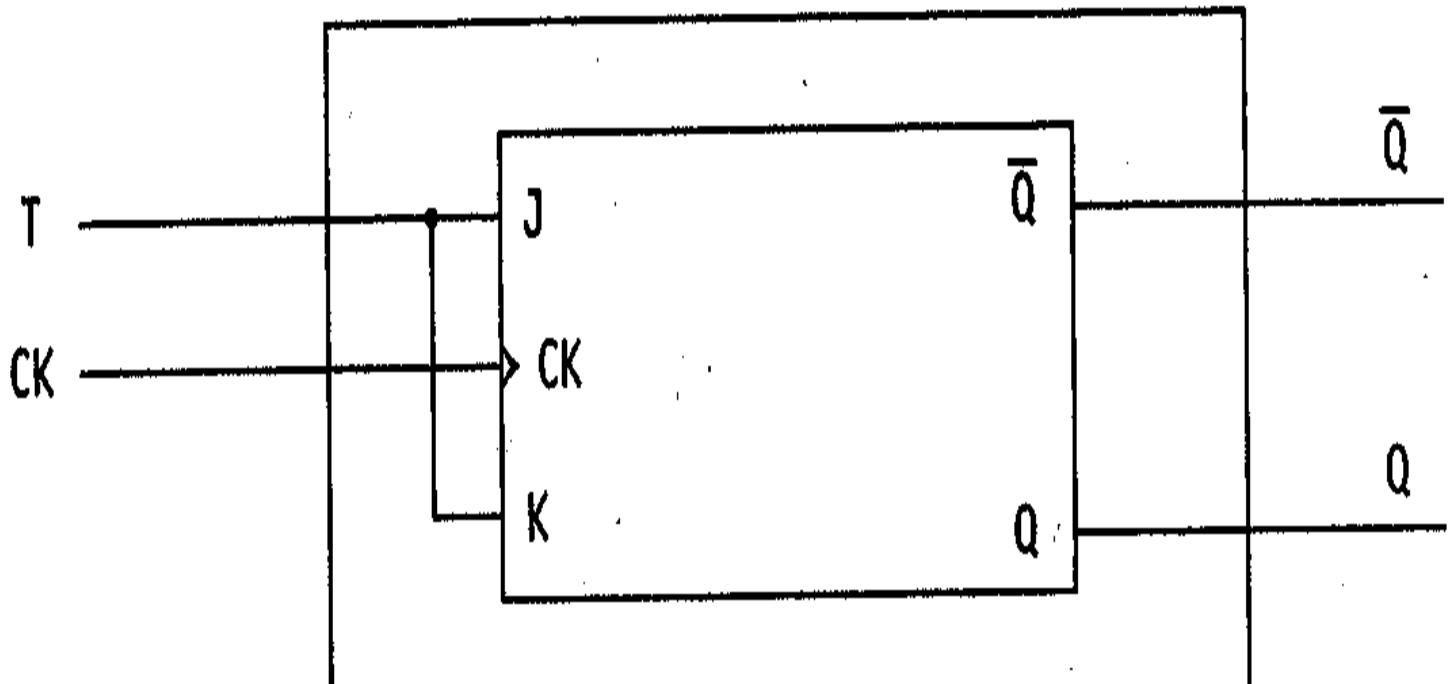


Positive edge triggered



If T = 1, circuit changes state on every positive edge of CK

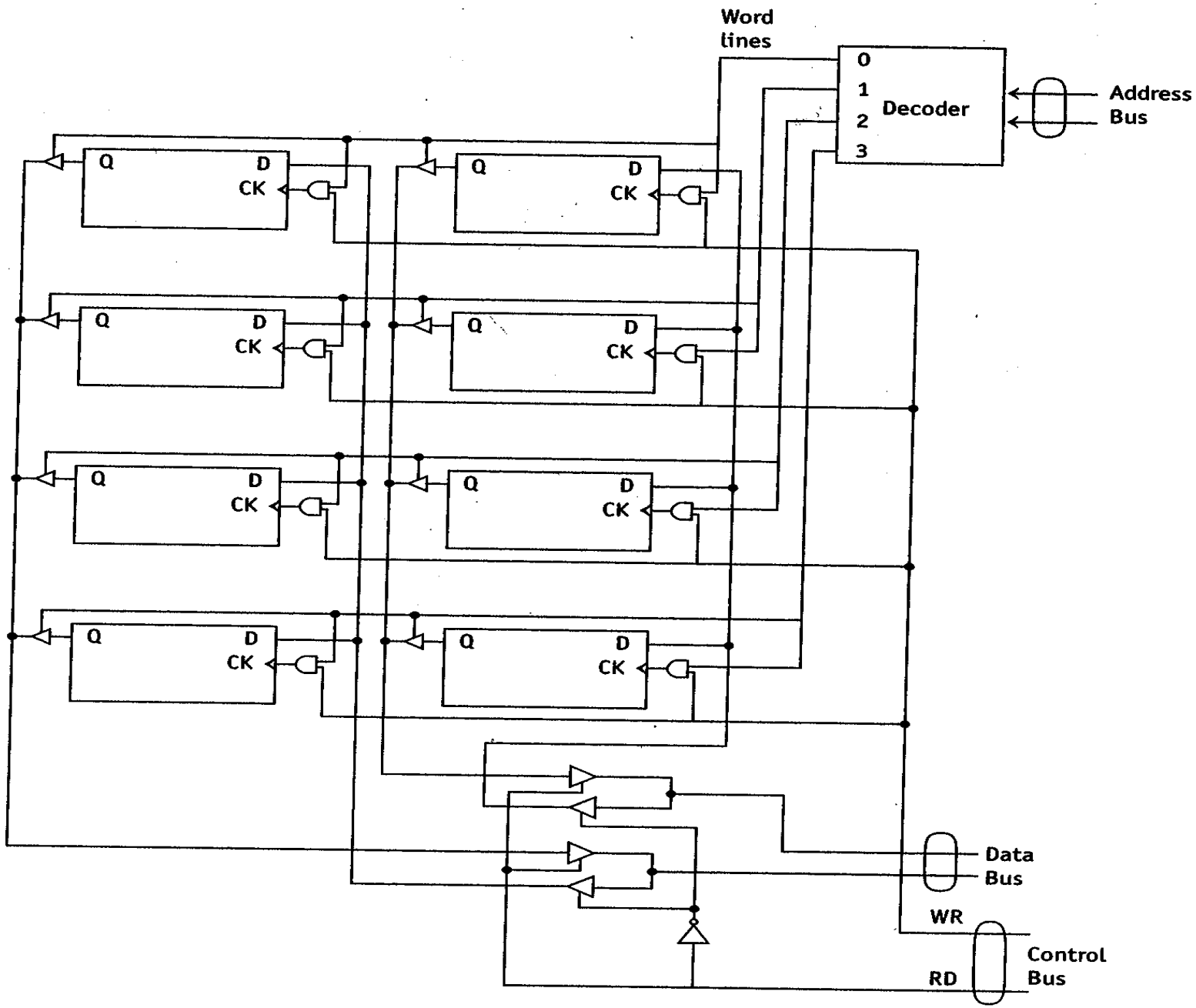
FIGURE 5.44 T Flip-Flop



We can construct RAM with edge-triggered D flip-flops and a decoder.

FIGURE 5.45

4 ◊ 2 RAM



Synchronous circuit:
All changes occur simultaneously.

Asynchronous circuit:
One subcircuit triggers the next.

FIGURE 5.46

4-Bit Binary Counter
(Asynchronous)

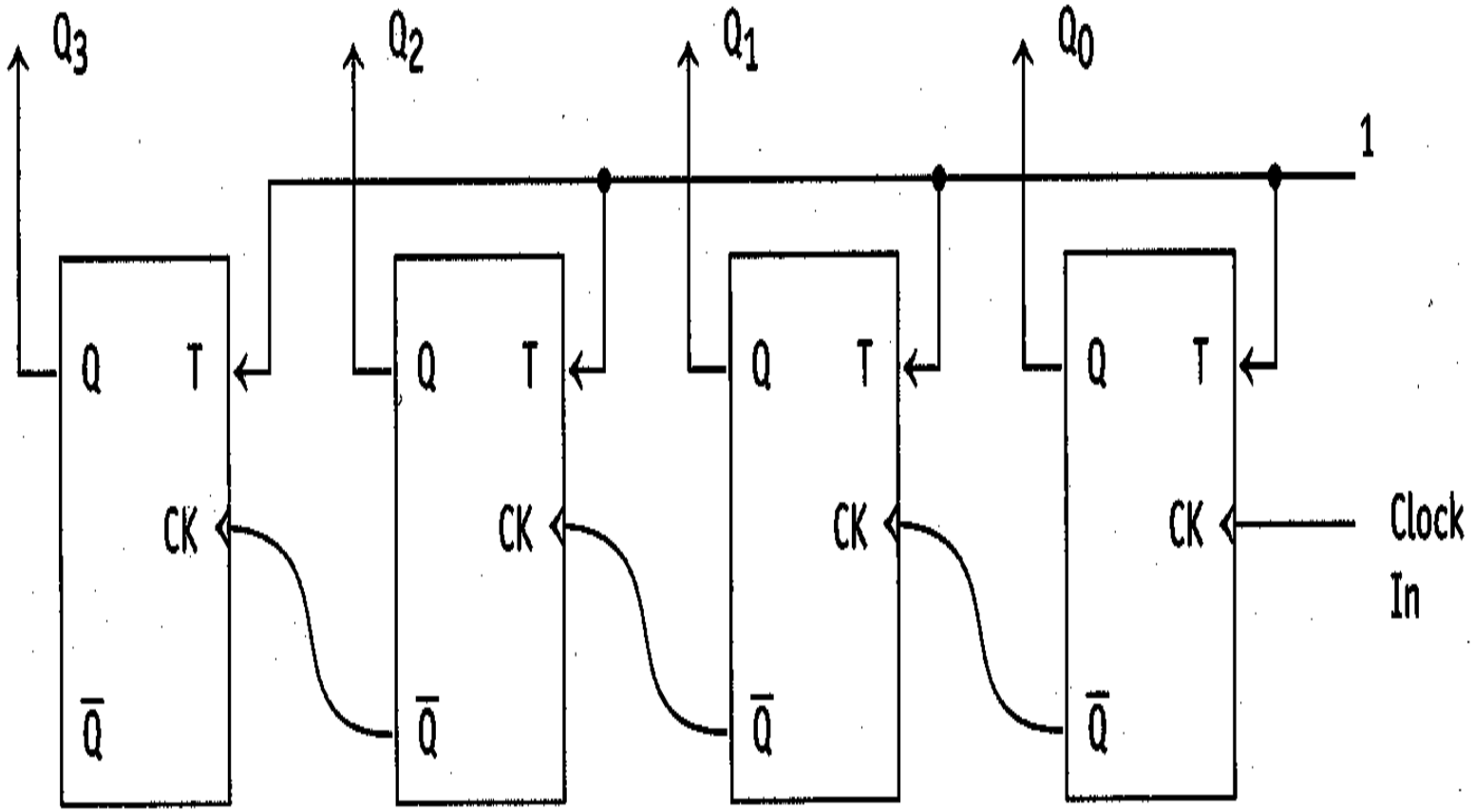


FIGURE 5.47

Read/Write Register
(Synchronous)

a)

Implementation of 4-bit register

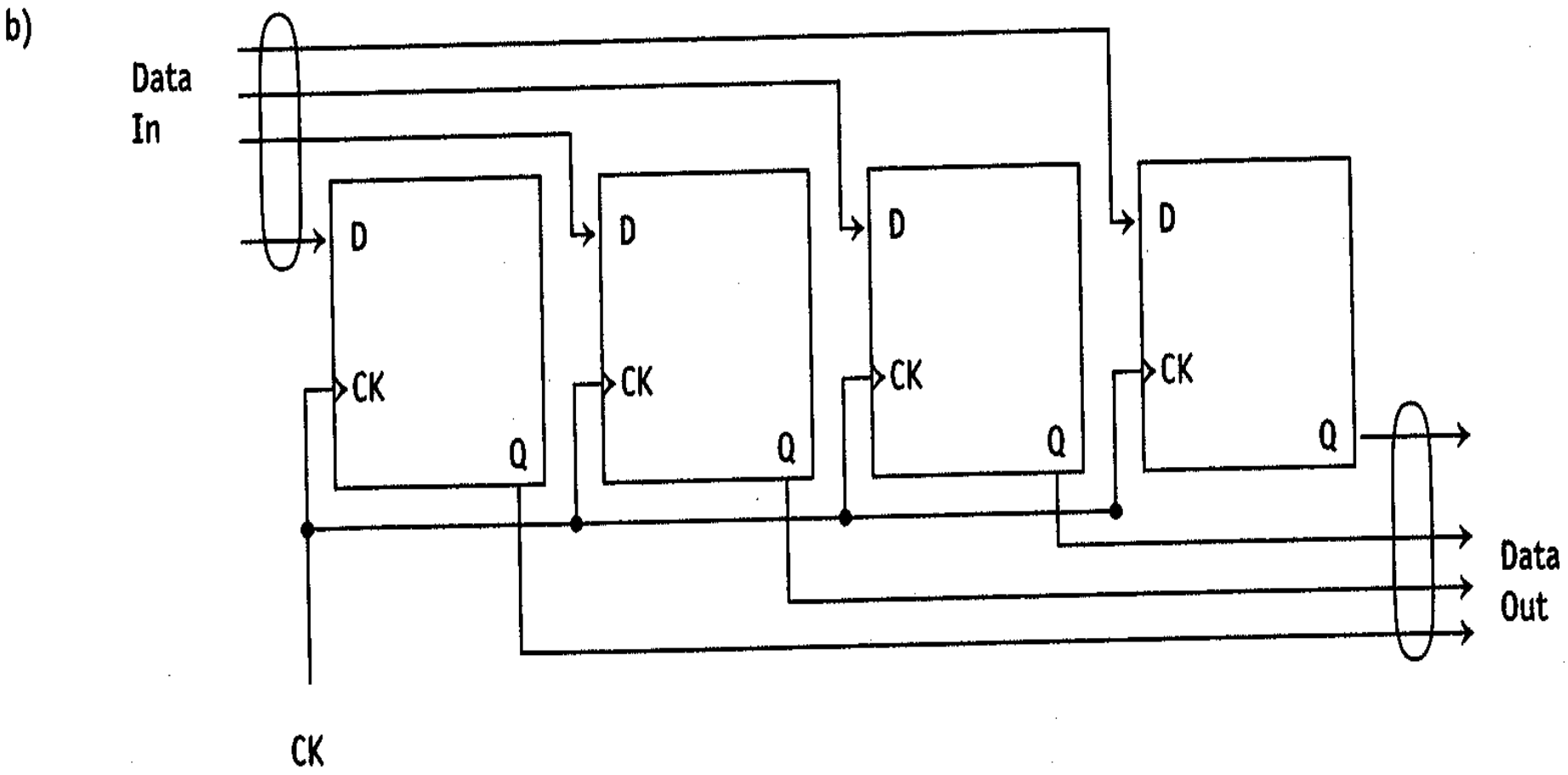
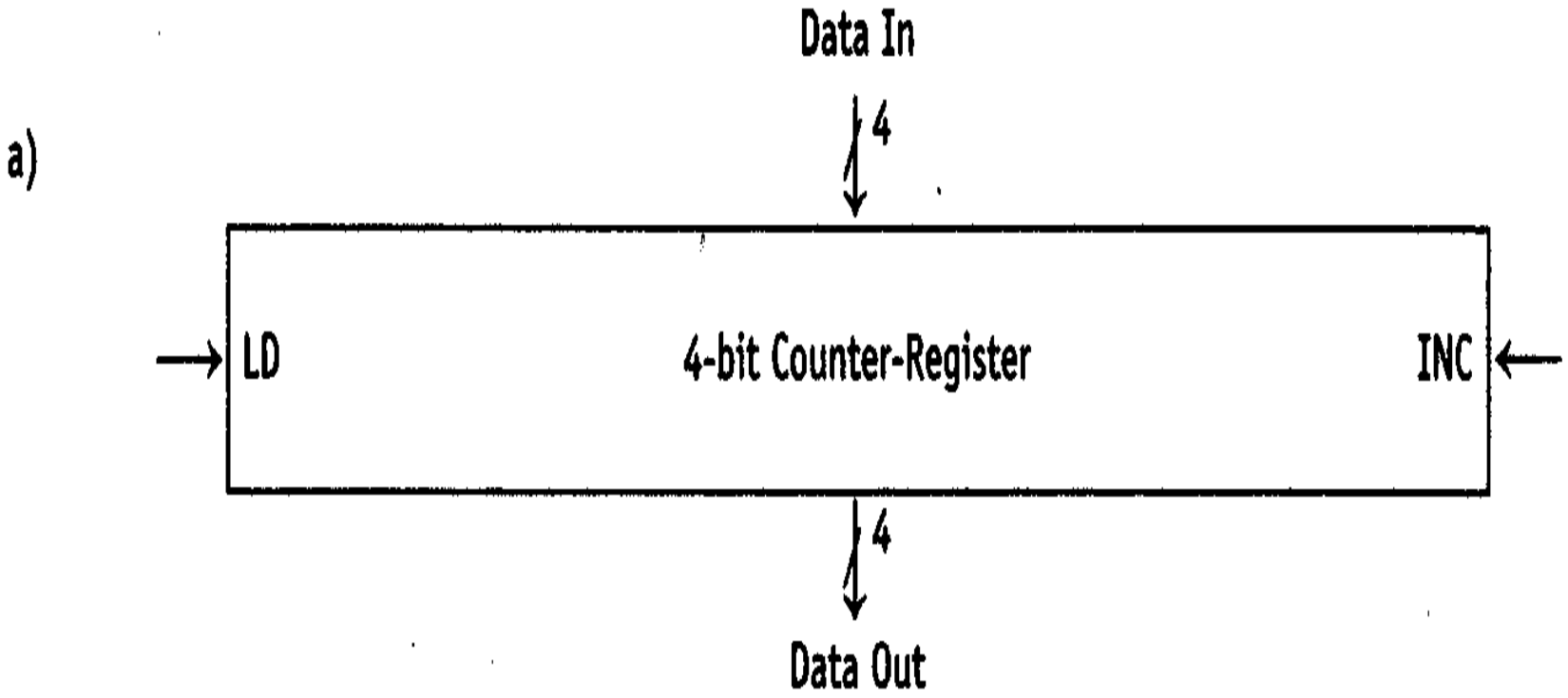


FIGURE 5.48 Counter-Register

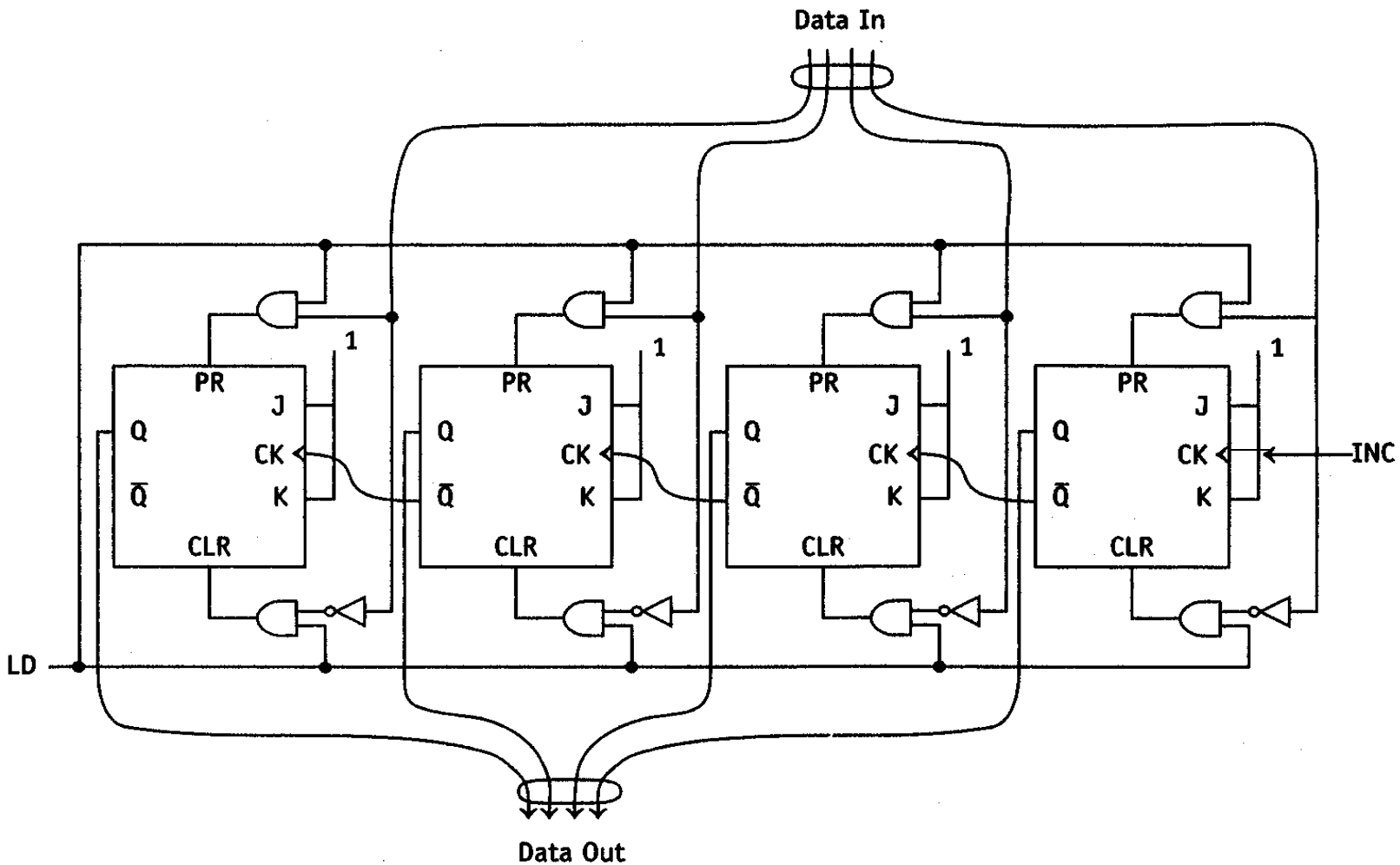


LD = 1: Register is loaded from Data In (not edge triggered).

INC = 1: Register is Incremented on leading edge.

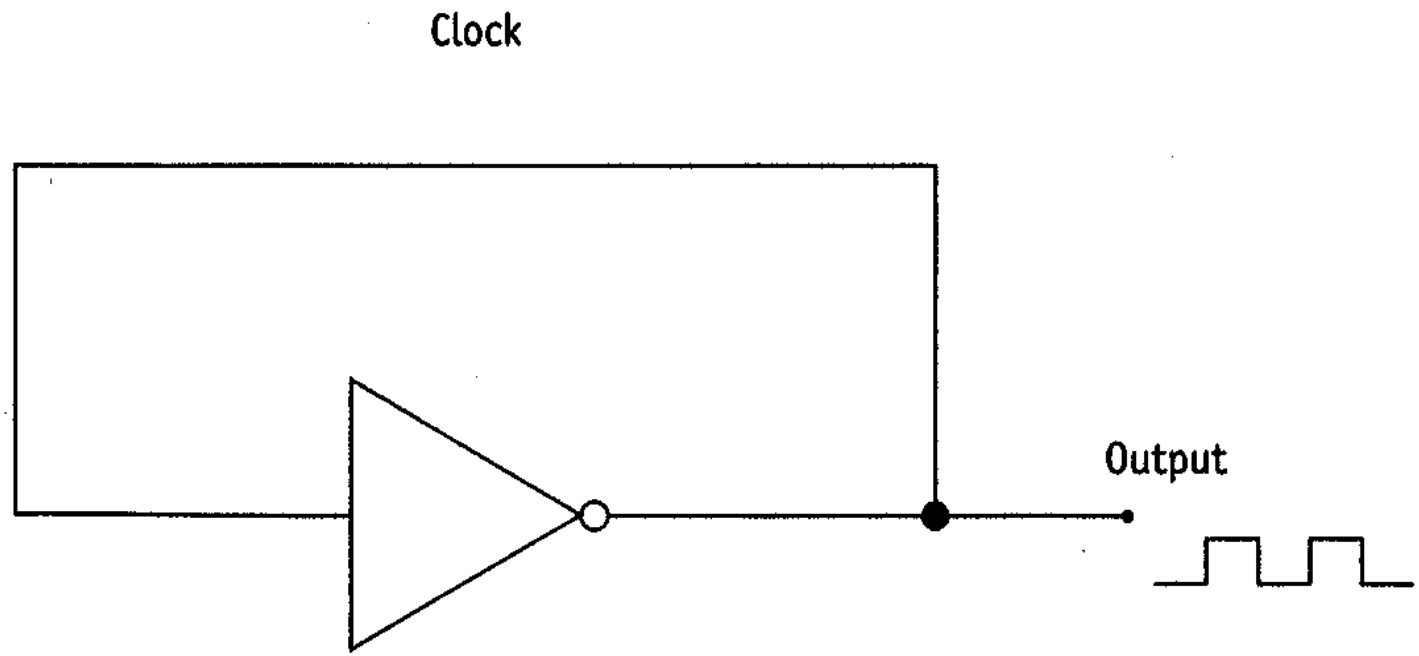
Implementation of counter-register

b)



A clock but not a good one.
A quartz crystal oscillator is better

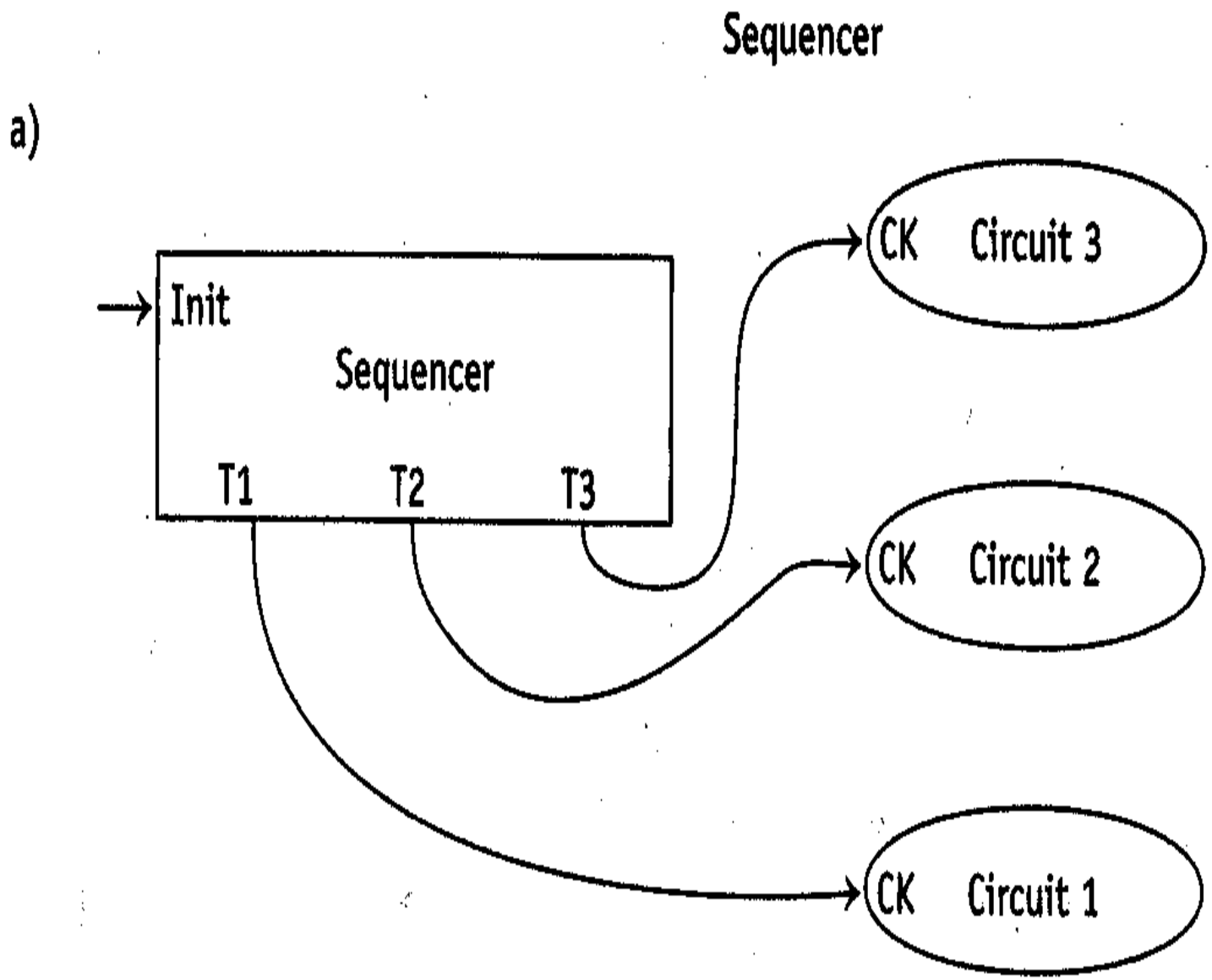
FIGURE 5.49



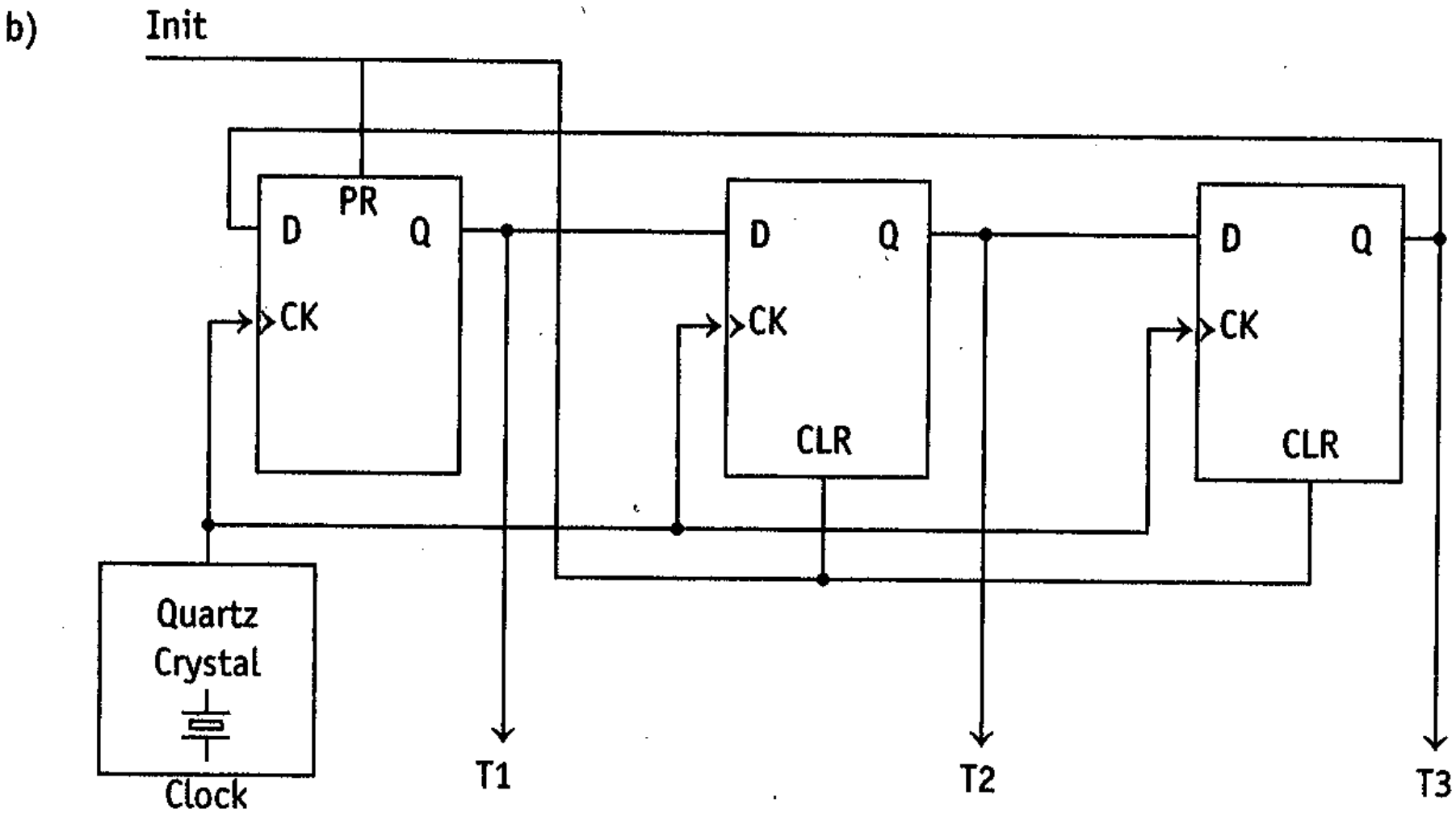
Quartz crystal

- Piezoelectric effect
- Used to create very frequency-stable clocks

FIGURE 5.50


Sequencer
a)
Init
Sequencer
T1
T2
T3
CK
Circuit 3
CK
Circuit 2
CK
Circuit 1

Implementation of sequencer



T1, T2, T3 clock subcycles